Electronic Colloquium on Computational Complexity, Report No. 29 (2005)

Speeding Up Approximation Algorithms for NP-hard Spanning Forest Problems by Multi-objective Optimization

Frank Neumann [⋆] and Marco Laumanns [⋆⋆]

Abstract. We give faster approximation algorithms for the generalization of two NP-hard spanning tree problems. First, we investigate the problem of minimizing the degree of minimum spanning forests. The task is to compute for each number of connected components a minimum spanning forest whose degree is as small as possible. Fischer [4] has shown how to compute a minimum spanning tree of degree at most $b \cdot \Delta^* + \lceil \log_b n \rceil$ in time $O(n^{4+1/\ln b})$ for any $b > 1$, where Δ^* is the value of an optimal solution. We model our generalization as a multi-objective optimization problem and give a deterministic algorithm that computes for each number of connected components a solution with the same approximation quality as the algorithm of Fischer and runs in time $O(n^{3+1/\ln b})$. After that, we take a multi-objective view on the problem of computing minimum spanning trees with nonuniform degree bounds, which has been examined by Könemann and Ravi [10]. Given degree bounds B_v for each vertex $v \in V$, we construct an algorithm that computes for each number of connected components a spanning forest in which each vertex v has degree $O(B_v + \log n)$ and whose weight is at most a constant times the weight of a minimum spanning forest obeying the degree bounds. The total runtime of our algorithm is $O(n^5)$, which is by a factor of $n \log n$ less than the algorithm of Könemann and Ravi.

1 Introduction

In this paper we consider two NP-hard spanning forest problems. Given a connected graph $G = (V, E)$ with n vertices and m edges and positive integer weights $w(e)$ for each edge $e \in E$, we are searching (i) for minimum spanning forests of minimum degree, and (ii) for minimum spanning forests obeying given degree bounds on the vertices. A forest with i connected components is a cycle-free subgraph of G that contains exactly $n - i$ edges. A minimum spanning forest with i connected components is a forest where the sum over all edge weights is minimal among all spanning forests with i connected components. In a minimum spanning forest of minimum degree, the largest vertex degree is as small

⋆ Institut of Computer Science, Christian-Albrechts-Univ. of Kiel, 24098 Kiel, Germany, email: fne@informatik.uni-kiel.de

⋆⋆ Institute of Operations Research, ETH Zürich, CH-8092 Zürich, Switzerland, email: laumanns@ifor.math.ethz.ch

ISSN 1433-8092

as possible. This generalizes the problem of computing minimum spanning trees of minimum degree.

The problem of computing minimum spanning trees is one of the fundamental problems in computer science and can be solved in polynomial time by greedy algorithms. The well-known algorithms due to Kruskal and Prim have worst-case runtimes of $O((n+m)\log n)$ and $O(n^2)$, respectively (see, e.g., Cormen, Leiserson, Rivest, and Stein [3]). The problem of constructing spanning subgraphs that satisfy given constraints has attained a lot of attention (see, e.g., [1, 2]). In most cases, even simple constraints lead to NP-hard problems (see Garey and Johnson [8]). The problem of minimizing the degree of spanning trees is NP-hard [8] as a spanning tree of degree 2 is a Hamiltonian path. This problem has a lot of application in the area of network design.

1.1 Previous work and our results

Let Δ^* be the maximum degree of an optimal solution. When edge weights are not considered, or assumed uniform, a $\Delta^* + 1$ approximation algorithm for minimizing the degree of spanning trees has been obtained by Fürer and Raghavachari [7]. For the weighted case, Fischer [4] has presented an approximation algorithm that computes a minimum spanning tree of degree at most $b \cdot \Delta^* + \lceil \log_b n \rceil$ in time $O(n^{4+1/\ln b})$ for any $b > 1$, which is the best-known algorithm for this problem up to now. His algorithm is an adaptation of a local search algorithm of Fürer and Raghavachari [6] to the weighted case. The idea of the local search is to perform edge exchanges until the spanning tree is locally optimal. The crucial point for bounding the runtime is the number of improvements that have to be executed until the approximation guarantee can be given. Fischer has shown, by a similar potential function argument as Fürer and Raghavachari [6], that the number of improvements is bounded by $O(n^{2+1/\ln b})$.

Könemann and Ravi [10] have considered the problem of approximating minimum spanning trees with nonuniform degree bounds. Given degree bounds B_v for all vertices, they have presented a combination of the primal-dual method and local search to compute a spanning tree in which the degree of each vertex v is $O(B_v + \log n)$ and the weight is by at most a constant factor higher than the weight of any spanning tree that obeys the given degree constraints. Their algorithm runs in time $O(n^6 \log n)$ and the analysis also uses the potential function argument given in [6].

We model the problem of computing minimum spanning forest of minimum degree as a multi-objective optimization problem where one objective is to minimize the number of connected components and the other objective to minimize the weight and degree. Our aim is to compute for each i, $1 \leq i \leq n$, a minimum spanning forest with i connected components that has the same approximation quality as the algorithm of Fischer. We show that the total number of local improvements can be bounded by $O(n^{1+1/\ln b})$, which is by a factor of n smaller than in the algorithm of Fischer. Based on this analysis we are able to construct a deterministic algorithm that computes a set of solutions containing a minimum spanning forest for each number of connected components with the same approximation guarantee as provided by Fischer's algorithm in time $O(n^{3+1/\ln b})$. Our algorithm can be seen as extension of Kruskal's algorithm for the computation of minimum spanning trees. Note that during the run, Kruskal's algorithm computes solutions that are minimum spanning forests for each possible number of connected components. The working principle of our algorithm is also to start with an empty graph and to compute the minimum spanning forests as in the run of Kruskal's algorithm one after another. After a new edge has been introduced that leads to a minimum spanning forest with smaller number of connected components, the degree of this minimum spanning forest is improved by edge exchanges as long as we cannot guarantee our desired approximation quality.

In the case of computing minimum spanning forests obeying given degree bounds we take a similar view. Starting with the empty edge set, we compute for each number of connected components a minimum spanning forest in which each vertex degree is $O(B_v \log n)$ in time $O(n^5)$. This beats the runtime of the algorithm given in [10] for the computation of minimum spanning trees with nonuniform degree bounds by a factor $n \log n$. Our algorithm uses an extension of the primal-dual approach given in [10] and our multi-objective view on the problem, which leads to a better bound on the number of local improvements.

The paper is organized as follows. In Section 2, we introduce our model for the computation of minimum spanning forests with minimum degree and give a new algorithm for minimizing the degree of minimum spanning forest that runs in time $O(n^{3+1/\ln b})$. Section 3 applies our technique in combination with an extension of the primal-dual method for minimum spanning trees [10] to the problem of computing minimum spanning forest with nonuniform degree bounds. We finish with some conclusions.

2 Minimizing the Degree of Minimum Spanning Forests

2.1 A Multi-objective Formulation

We take a multi-objective view on the computation of minimum spanning forests with minimum degree. Let $X = \{0,1\}^m$ be the search space. A search point $x \in X$ describes the set of all edges e_i where $x_i = 1$ holds. Let $c(x)$ be the number of connected components of the solution x, $w(x)$ be the weight of the chosen edges, $d_j(x)$ be the number of vertices of degree j in x, and $\Delta(x)$ the maximum vertex degree of x. The value of x is given by the vector $f(x) = (f_1(x), f_2(x))$, where $f_1(x) = c(x)$ and $f_2(x) = (w(x), d_{n-1}(x), \ldots, d_0(x))$. Both objectives f_1 and f_2 have to be minimized. Minimizing the second objective means minimization with respect to the lexicographic order. This model generalize the function $g(x) = (c(x), w(x))$ that has been examined by Neumann and Wegener [11] for the computation of minimum spanning trees by randomized search heuristics. They have shown that a multi-objective view leads to a more efficient optimization process than in the case of a single objective one.

Let $f(X)$ be the image of the search space under the objective function f defined above. By intersecting the canonic order on $f_1(X)$ with the lexicographic order on $f_2(X)$, both of which are total orders, a partial order on $f(X)$ can be defined as

$$f(x) \preceq f(x') :\Leftrightarrow f_1(x) \leq f_1(x) \wedge f_2(x) \leq_{\text{lex}} f_2(x')$$

for all $x, x' \in X$. This partial order represents our preference relation regarding the solutions. The aim is to identify all minimal elements of $(f(X), \preceq)$, and with each minimal element one of its pre-images from X.

As the edge weights are positive, a minimum spanning forest with i connected components has a smaller weight than a minimum spanning forest with $i+1$ connected components. Therefore, $(f(X), \preceq)$ has n minimal elements, representing for each $i \in \{1, \ldots, n\}$ a minimum spanning forest with i connected components and minimum degree. Our goal is to approximate the set of minimal elements as good as possible. We want to compute for each i, $1 \leq i \leq n$, a minimum spanning forest with i connected components that has degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$, where Δ_i^* is the smallest maximum degree of any minimum spanning forest with i connected components.

2.2 Local Improvements

Fischer's algorithm [4] for the computation of a minimum spanning tree with degree at most $b \cdot \Delta^* + \lceil \log_b n \rceil$ starts with an arbitrary minimum spanning tree and improves the degree of high-degree vertices. The number of these improvements is bounded by $O(n^{2+1/\ln b})$. A better bound on the number of necessary improvements would yield a better upper bound for Fischer's algorithm. We consider the number of necessary improvements for our multi-objective model and start with some general properties of minimum spanning forests with i connected components.

Lemma 1. *Let Δ_i^*, $1 \leq i \leq n$, be the minimum degree of a minimum spanning forest with i connected components. Then $\Delta_n^* \leq \Delta_{n-1}^* \leq \ldots \leq \Delta_1^*$ holds.*

Proof. Suppose that $\Delta_i^* > \Delta_{i-1}^*$ holds for some $i \in \{2, \ldots n\}$. Let F_{i-1}^* be a minimum spanning forest with $i-1$ connected components and minimum degree. Then we can delete the heaviest edge from F_{i-1}^* to construct a minimum spanning forest with i connected components whose degree is at most Δ_{i-1}^*. This contradicts the assumption. □

Let s_i be a solution with i connected components and minimal weight. We call s_i locally optimal if there is no solution s_i' with $c(s_i') = c(s_i)$ and hamming distance 2 that is better than s_i with respect to $f_2(s_i)$ when disregarding all $d_j(s_i)$ with $j < \Delta(s_i) - \lceil \log_b n \rceil$. Otherwise, we say that s_i' improves s_i. For the case $i = 1$ Fischer has shown in [4] that if there is no improvement for s_1 then the minimum spanning tree has already degree at most $b \cdot \Delta_1^* + \lceil \log_b n \rceil$ for any $b > 1$. We generalize this approximation guarantee of local optimal minimum spanning trees given by Fischer to locally optimal minimum spanning forests.

Lemma 2. *Let s_i be a solution that is locally optimal and Δ_i be its maximum degree. Then $\Delta_i \leq b \cdot \Delta_i^* + \lceil \log_b n \rceil$ holds for any constant $b > 1$.*

Proof. Consider a locally optimal forest F described by s_i. Let U_i be the set of vertices of degree at least i in F. The number of vertices in U_i is at most n for each i. Hence, the ratio $\frac{|U_{i-1}|}{|U_i|}$ cannot be greater than b for $\log_b n$ consecutive values of i.

Consider a δ with $\Delta_i - \lceil \log_b n \rceil \leq \delta \leq \Delta_i$ such that $\frac{|U_{\delta-1}|}{|U_\delta|} \leq b$. Deleting all edges from F that are adjacent to vertices of U_δ yields a forest F_δ with at least $(\delta - 1)|U_\delta| + 1 + i$ connected components. As F is locally optimal, only edges adjacent to vertices of $U_{\delta-1}$ may participate in a minimum spanning forest with i connected components. Hence, there must be at least $(\delta - 1)|U_\delta| + 1 \geq (\delta - 1)(|U_{\delta-1}|/b) + 1$ edges adjacent to vertices of $U_{\delta-1}$. The average degree of a vertex in $U_{\delta-1}$ is therefore at least $\frac{(\delta-1)|U_{\delta-1}|+b}{b \cdot |U_{\delta-1}|}$, which implies $\Delta_i^* > \frac{\delta-1}{b}$. Using $\delta \geq \Delta_i - \lceil \log_b n \rceil$ we get $\Delta_i \leq b \cdot \Delta_i^* + \lceil \log_b n \rceil$. □

Lemma 3. *The total number of local improvements until a minimum spanning forest of degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$ has been computed for each i, $1 \leq i \leq n$, can be bounded by $O(n^{1+1/\ln b})$.*

Proof. Consider a situation where a minimum spanning forest with j connected components and degree at most $b \cdot \Delta_j^* + \lceil \log_b n \rceil$ has been computed for each j, $i \leq j \leq n$. We prove by induction on i that the number of improvements steps to reach this situation is bounded by $(n - i) \cdot O(n^{1/\ln b})$. The empty edge set $s_n = \emptyset$ is obviously a minimum spanning forest with n components and minimum degree, and no improvements are necessary to reach this. For going from i to $i-1$ we introduce into s_i a lightest edge e that does not create a cycle. This yields a minimum spanning forest with $i - 1$ components. If e is not adjacent to at least one vertex of degree $\Delta(s_i)$ in s_i, the equality $\Delta(s_{i-1}) = \Delta(s_i)$ holds and s_i is a minimum spanning forest with degree at most $b \cdot \Delta_{i-1}^* + \lceil \log_b n \rceil$ due to Lemma 1. Otherwise, we calculate the number of improvements that are necessary until s_{i-1} is locally optimal or $\Delta(s_{i-1}) = \Delta(s_i)$ holds. Consider the potential function $g(s) := \sum_{j=0}^{\lceil \log_b n \rceil + 1} d_{r+j}(s) \cdot e^j$, where $r = \Delta(s_i) - \lceil \log_b n \rceil$. Introducing an arbitrary edge into s_i increases the potential by at most $2e^{\lceil \log_b n \rceil + 1} \leq 2e^2 \cdot e^{\log_b n} = 2e^2 \cdot e^{\ln n \cdot \log_b e} = O(n^{1/\ln b})$. A local improvement reduces the potential by at least $e^2 - 3e + 2$, i.e., a constant amount because $d_k(s)$ decreases by at least 1 for some k and, in the worst case, $d_l(s)$ increases by three but l must be strictly smaller than k to qualify for an improvement. Therefore, introducing an arbitrary edge into s_i increases the potential by an amount that can be compensated by $O(n^{1/\ln b})$ improvements as long as s_{i-1} is not locally optimal. Reducing the potential by $O(n^{1/\ln b})$ improvements leads to a solution s_{i-1} with $\Delta(s_{i-1}) \leq \Delta(s_i)$. If we have reached a locally optimal solution within these improving steps a solution with degree at most $b \cdot \Delta_{i-1}^* + \lceil \log_b n \rceil$ has already been achieved due to Lemma 2. This proves the claim of the induction. Observing that the smallest value for i is 1 completes the proof. □

```
1. Initialize: let $i := n$, $s_i := \emptyset$, $S := \{s_i\}$.
2. Create $s_{i-1}$ by introducing into $s_i$ the lightest edge that does not create a cycle.
3. Improve the solution $s_{i-1}$ until it is locally optimal or $\Delta(s_{i-1}) = \Delta(s_i)$ holds.
4. $S := S \cup \{s_{i-1}\}$
5. $i := i - 1$
6. If $i > 1$ continue at 2., otherwise output $S$ and stop.
```

Fig. 1. Minimum Spanning Forest Optimizer (MSFO)

2.3 The Algorithm

The analysis in Section 2.2 has shown that the number of improvements in the multi-objective model can be bounded by $O(n^{1+1/\ln b})$, which is by a factor n less than then the number of improvements in the algorithm of Fischer. Based on this observation we give a deterministic algorithm that computes for each i a minimum spanning forest with i connected components and degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$ in time $O(n^{3+1/\ln b})$ for any $b > 1$.

Let s_i be a minimum spanning forest of degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$. Then we can produce a minimum spanning forest s_{i-1} with $i-1$ connected components by introducing a lightest edge that does not create a cycle. If $\Delta(s_{i-1}) = \Delta(s_i)$ holds, s_{i-1} has the desired approximation quality. Otherwise we have to improve s_{i-1}. The pseudo-code of our algorithm called Minimum Spanning Forest Optimizer (MSFO) is given in Figure 1.

MSFO can be seen as a variant of Kruskal's algorithm where after each insertion of an edge the degree of the current solution s_{i-1} is improved as long as we cannot guarantee the desired approximation quality. The algorithm of Kruskal can be implemented in time $O((m + n) \log n)$. Hence, to bound the runtime of MSFO it is necessary to bound the number of local improvements (as done in Section 2.2) and the time to achieve such an improvement.

Lemma 4. *Let s_i be a solution with i connected components that is not locally optimal. Then an improvement can be found in time $O(n^2)$.*

Proof. There are two possibilities to improve a solution s_i. Let F be the corresponding minimum spanning forest. In the first case, the introduced edge connects two components of F and an edge from the forest has to be removed. In the second case, the improvement is achieved by introducing an edge e that creates a cycle in F. Then an edge from this cycle has to be deleted to create a new spanning forest with i connected components.

Let $W_1, \ldots, W_k$ be the distinct weight classes for which there are edges in F. For the first case, consider the edges of W_j for each j, $1 \leq j \leq k$, in F and compute the edge that reduces the value of the potential function of Lemma 3 by the largest amount. This computation can be done in time $O(n)$ for all weight classes because the computation of the reduction for a single edge e can be done in constant time and there are at most $n - 2$ edges to consider. Let $g_{\min}^j$ be the smallest value that can be obtained by removing an edge e_j of weight class W_j

from F. Now we consider each edge $e' \in E \setminus F$ of weight class W_j and introduce e' if the result improves s_i. We consider in the process each edge at most once and the computation of the desired potential difference can be implemented in constant time. Hence, an improvement can be found in time $O(n^2)$ in this case.

For the second case we use the idea of Fischer and investigate a depth first search traversal of the forest F represented by s_i from every vertex $v \in V$. Let w be the current vertex of the traversal and P_w be the set of edges on the path from v to w. We assign variables $M_1, \ldots, M_k$ such that $M_j, 1 \leq j \leq k$, denotes the maximum degree of those vertices adjacent to edges of weight class W_j in P_w. For a depth first search traversal we can compute the M_i variables in constant time per step using stacks. If there is an edge $(v, w) \in E$, let w_i be its weight. If M_i is at least two greater than the degree of v and w, and M_i is at least $\Delta(s_i) - \lceil \log_b n \rceil$, then adding (v, w) to s_i and deleting some edge from P_w of weight class W_i adjacent to a vertex of degree M_i constitutes an improvement. The computation of the n depth first search traversals can be carried out in time $O(n^2)$ which completes the proof. □

Using the bound on the number of necessary improvements and the time bound to achieve such an improvement, we can give an upper bound on the runtime of MSFO.

Theorem 1. *The algorithm MSFO computes for any $b > 1$ in time $O(n^{3+1/\ln b})$ a set of solutions that includes for each $i \in \{1, \ldots, n\}$ a minimum spanning forest with i connected components and degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$.*

Proof. Consider the time the solutions $\{s_n, s_{n-1}, \ldots, s_i\} \subset S$ have been produced. These solutions have the following properties. Each s_j, $i \leq j \leq n$, is a minimum spanning forest with j connected components. In addition, s_j is locally optimal or $\Delta(s_j) = \Delta(s_{j+1})$ holds. Obviously, s_n is a locally optimal solution. We introduce into s_i an edge e of minimal weight that does not create a cycle. This can be easily done by checking each remaining edge in time $O(m)$. Note that the whole computation in step 2 in the run of the algorithm can be implemented in time $O((m+n) \log n)$ using the ideas of Kruskal's algorithm. After step 3, the solution s_{i-1} has minimal weight among all solutions with $i-1$ components. If e is not incident to at least one edge of degree $\Delta(s_i)$, $\Delta(s_{i-1}) = \Delta(s_i)$ holds. Otherwise, the number of vertices with degree $\Delta(s_i) + 1$ is at most 2 and we have to improve s_{i-1} to reach a locally optimal solution or to achieve $\Delta(s_{i-1}) = \Delta(s_i)$.

The number of local improvements in the run of MSFO is $O(n^{1+1/\ln b})$ as shown in Lemma 3 and an improvement of a non locally optimal solution can be found in time $O(n^2)$ due to Lemma 4. Hence, the time until MSFO has achieved the desired approximation can be bounded by $O(n^{3+1/\ln b})$. □

3 Minimum Spanning Forests with Nonuniform Degree Bounds

Könemann and Ravi [10] have examined the case of non-uniform degree bounds B_v for all vertices $v \in V$. They presented an algorithm that finds, in time

$O(n^6 \log n)$, a spanning tree where the degree of each vertex is $O(B_v + \log n)$ and whose total edge weight is at most a constant times the weight of any tree that satisfies the degree constraints. The algorithm uses a combination of primal-dual methods and local search, where in each local search step the normalized degree of the high-degree vertices in a current spanning tree is reduced. We generalize the primal-dual idea of Könemann and Ravi to the approximation of minimum spanning forests with nonuniform degree bounds. The task is to find for each i, $1 \leq i \leq n$, a spanning forest with i connected components and minimum total edge weight such that the maximum degree of each vertex v is at most B_v. The algorithm presented here runs in time $O(n^5)$ and outputs for each i, $1 \leq i \leq n$, a spanning forest F_i of i connected components whose vertex degrees are $O(B_v + \log n)$ and whose total weight is at most a constant times the total weight of any minimum spanning forest with i connected components.

We first adapt some results of [10] to the case of spanning forests. A feasible partition of V is a set $\pi = \{V_1, \ldots, V_k\}$ where $V_i \cap V_j = \emptyset$ for all $i \neq j$, $V = V_1 \cup \ldots \cup V_k$, and the induced subgraphs $G[V_i]$ are connected. Let G_π be the graph obtained from G by contracting each V_i into a single vertex, Π be the set of all feasible partitions of V,and $x(e)$ be the variable indicating whether edge e is included in the current solution, i.e., $x(e_i) = x_i$. We consider the following integer linear program (IP) formulation for the problem of computing the minimum spanning forest with i, $1 \leq i \leq n$, connected components that obeys all degree bounds B_v.

$$\min \sum_{e \in E} w(e)x(e) \tag{1}$$

$$\text{s.t.} \sum_{e \in E[G_\pi]} x(e) \geq |\pi| - i \quad \forall \pi \in \Pi \tag{2}$$

$$\sum_{e \in E : v \in e} x(e) \leq B_v \quad \forall v \in V \tag{3}$$

$$x(e) \in \{0,1\} \quad \forall e \in E \tag{4}$$

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

$$\max \sum_{\pi \in \Pi} (|\pi| - i) \cdot y_\pi - \sum_{v \in V} \lambda_v B_v \tag{5}$$

$$\text{s.t.} \sum_{\pi : e \in E[G_\pi]} y_\pi \leq w(e) + \lambda_u + \lambda_v \quad \forall e = (u,v) \in E \tag{6}$$

$$y, \lambda \geq 0 \tag{7}$$

Könemann and Ravi have given a primal-dual interpretation of Kruskal's algorithm. Let (IP-SP) denote (IP) without constraints of type (3) its LP relaxation denoted by (LP-SP) and its dual be (D-SP). Kruskal's algorithm can be seen as a continuous process over time that starts with an empty edge set at time 0 and ends with a minimum spanning tree at time t^*. At any time t,

$0 \le t \le t^*$ a pair (x^t, y^t) is kept, where x^t is a partial primal solution for (LP-SP) and y^t is feasible solution for (D-SP). In the initialization step $x(e)^0 = 0$ is set for all $e \in E$, and $y^t_\pi = 0$ for all $\pi \in \Pi$. Consider the forest F^t that corresponds to the partial solution x^t and let π^t be the partition induced by the connected components of $G[F^t]$. At time t the algorithm increases y^t_π until a constraint of type (6) becomes tight. If this happens for edge e, this edge e is included into the primal solution. If more than one edge becomes tight, the edges are processed in arbitrary order. We denote by MSF_i a variant of this algorithm that stops when a minimum spanning forest with i connected components has been computed in the continuous process.

Let $deg_F(v)$ be the degree of vertex v in the spanning forest F with i connected components. The normalized degree of a vertex v is denoted by $ndeg_F(v) = \max\{0, deg_T(v) - 1 - b\alpha \cdot B_v\}$, where b and α are constants depending on the desired approximation quality. Let Δ^t the maximum normalized degree of any vertex in the current spanning forest F^t_i at a given time t and denote by U^t_j the set of vertices whose normalized degree is at least j at time t. The following lemma was shown in [10].

Lemma 5. *There is a* $d^t \in \{\Delta^t - 2\log_b n, \ldots, \Delta^t\}$ *such that*

$$\sum_{v \in U_{d^t-1}} B_v \le b \cdot \sum_{v \in U_{d^t}} B_v$$

for any constant $b > 1$.

Proof. Suppose that for all $d^t \in \{\Delta^t - 2\log_b n, \ldots, \Delta^t\}$ the relation

$$\sum_{v \in U_{d^t-1}} B_v > b \cdot \sum_{v \in U_{d^t}} B_v$$

holds. We may assume $B_v \le n-1$ for any v, which implies $\sum_{v \in V} B_v \le n(n-1)$. Since there is at least one vertex of normalized degree Δ^t, we have

$$\sum_{v \in U_{\Delta^t - 2\log_b n}} B_v \ge b^{2\log_b n} = n^2,$$

a contradiction. □

Our algorithm called Primal Dual Forest Optimizer (PDFO) is given in Figure 2. The idea of the algorithm is to start with an empty edge set and compute the solutions with the desired approximation quality one after another. If we are considering a solution x^t with i connected components that does not have the desired approximation quality with respect to the degree bounds, we compute a new solution x^{t+1} which improves x^t with respect to the normalized degree. Let F^t_i be the forest corresponding to x^t. We increase the weight of an edge $e \in E$ by ϵ^t if it is either in F^t_i and adjacent to vertices of U_{d^t}, or in $E \setminus F^t_i$ and adjacent to vertices of U_{d^t-1}. The weight increment ϵ^t is defined as the smallest

```
1. t:=0; $\lambda_v^t := 0$, $\forall v \in V$; $w^t(e) = w(e)$, $\forall e \in E$;
2. $i := n$; $(x^t, y^t) := \mathrm{MSF}_i(G, w^t)$; $S := \{(x^t, y^t)\}$;
3. while $i > 1$ do
   (a) $i := i - 1$; $w^{t+1}(e) = w^t(e)$; $(x^{t+1}, y^{t+1}) := \mathrm{MSF}_i(G, w^{t+1})$; $t := t + 1$;
   (b) while $\Delta^t > 2 \log_b n$ do
       i. Choose $d^t \in \{\Delta^t - 2\log_b n, \ldots, \Delta^t\}$ s.t. $\sum_{v \in U^t_{j-1}} B_v \leq b \cdot \sum_{v \in U^t_j} B_v$
       ii. Choose $\epsilon^t$ and let $\lambda_v^{t+1} := \lambda_v^t + \epsilon^t$ if $v \in U^t_{d^t-1}$ and $\lambda_v^{t+1} := \lambda_v^t$ otherwise
       iii. $w^{t+1}(e) := w^t(s) + \epsilon^t$ if $((e \in F_i^t) \wedge (e \cap U_{d^t} \neq \emptyset) \vee ((e \notin F_i^t) \wedge (e \cap U_{d^t-1} \neq \emptyset))$
            and $w^{t+1}(e) := w^t(e)$ otherwise
       iv. $(x^{t+1}, y^{t+1}) := \mathrm{MSF}_i(G, w^{t+1})$; $t := t + 1$;
   (c) $S := S \cup \{x^t\}$;
```

Fig. 2. Primal Dual Forest Optimizer (PDFO)

weight increase when deleting an edge adjacent to a vertex of U_{d^t} and inserting an edge adjacent to vertices that are not contained in U_{d^t-1} such that a new cyclefree subgraph of G is constructed. After that, x^{t+1} is a minimum spanning forest with i connected components with respect to the updated weight function w^{t+1}. We have also stated the computation of the dual variables corresponding to the primal solutions in Figure 2 using the algorithm $\mathrm{MSF_i}$. The dual variables will be used later to show the approximation quality of our algorithm, but it is not necessary to carry out the computation of these variables in the run of the algorithm.

We want to show that the algorithm computes for each i, $1 \leq i \leq n$, a solution with i connected components in which each vertex v has degree $O(B_v + \log n)$ and weight at most a constant times the weight of an optimal solution obeying the degree bounds in time $O(n^5)$. First, we consider the approximation quality of the solutions that are introduced into the set S in step 3c. Here we use an extension of the arguments given in [10] to the case of minimum spanning forest with given degree bounds.

Lemma 6. *For all iterations $t \geq 0$ where we are considering solutions with i connected components in the algorithm PDFO, the relation*

$$\sum_{\pi \in \Pi} (|\pi| - j) y_\pi^{t+1} \geq \sum_{\pi \in \Pi} (|\pi| - j) y_\pi^t + \epsilon^t \alpha \sum_{v \in U_{d^t-1}} B_v \tag{8}$$

holds for and all j, $1 \leq j \leq i$.

Proof. We generalize the ideas in the proof of Claim 1 in [10]. Let $F_j^t = \{e_1^t, \ldots, e_{n-j}^t\}$ be the set of edges that would be produced by a run of the algorithm MSF_j in iteration t. The change of the dual objective function value in iteration t for a specific value of j is given by

$$\sum_{\pi \in \Pi} (|\pi| - j) \cdot (y_\pi^{t+1} - y_\pi^t) = \sum_{l=1}^{n-j} (r_l^{t+1} - r_l^t)$$

where r_l^t is the time at which the MSF$_j$ algorithm includes the edge e_l^t. Assume that we are considering solutions with i connected components in iteration t. Then we lengthen all edges $e \in F_i^t$ that are incident to vertices of normalized degree at least d^t. This implies that all these edges become tight ϵ^t time later. Using that all edges of F_i^t are also contained in each minimum spanning forest F_j^t for $j \leq i$, we get

$$\sum_{\pi \in \Pi} (|\pi| - j) \cdot (y_\pi^{t+1} - y_\pi^t) \geq \epsilon^t \cdot |E(U_{d^t}) \cap F_i^t|.$$

Here $E(U_{d^t})$ denotes the set of edges that are incident to vertices from U_{d^t}. F_i^t is a minimum spanning forest with i connected components. This implies that there are at most $|U_{d^t}| - i$ edges in $E(U_{d^t})$ that are incident to two vertices from U_{d^t}, therefore

$$\epsilon^t \cdot |E(U_{d^t}) \cap F_i^t| \;\geq\; \epsilon^t \cdot \left[\left(\sum_{v \in U_{d^t}} (b\alpha + 1/B_v) \cdot B_v \right) - (|U_{d^t}| - i) \right].$$

This leads to

$$\sum_{\pi \in \Pi} (|\pi| - j)(y_\pi^{t+1} - y_\pi^t) \;\geq\; \epsilon^t \alpha b \cdot \left(\sum_{v \in U_{d^t}} B_v \right) + i \geq \epsilon^t \alpha b \cdot \sum_{v \in U_{d^t}} B_v,$$

and using Lemma 5 we get

$$\sum_{\pi \in \Pi} (|\pi| - j)(y_\pi^{t+1} - y_\pi^t) \;\geq\; \epsilon^t \alpha \cdot \sum_{v \in U_{d^t - 1}} B_v,$$

which completes the proof. □

Lemma 7. *Let $\omega > 1$ be a constant and $\alpha = \max\{\omega/(\omega-1), \omega\}$. For all iterations $t \geq 0$ where we are considering solutions with i connected components in the algorithm PDFO, the relation*

$$\omega \sum_{v \in V} B_v \lambda_v^t \;\leq\; (\omega - 1) \sum_{\pi \in \Pi} (|\pi| - j) \cdot y_\pi^t \tag{9}$$

holds for each j, $1 \leq j \leq i$.

Proof. After initialization, $\sum_{v \in V} B_v \lambda_v^0 = \sum_{\pi \in \Pi} (|\pi| - j) \cdot y_\pi^0 = 0$ holds. Lemma 6 implies that the right hand side of (9) increases by at least $(\omega - 1) \cdot \alpha \epsilon^t \sum_{v \in U^t_{d^t - 1}} B_v$. The left hand side increases by $\omega \cdot \epsilon^t \sum_{v \in U^t_{d^t - 1}} B_v$. Using $\alpha \geq \omega/(\omega - 1)$, the relation is maintained. □

Lemma 8. *For all iterations $t \geq 0$ where we are considering solutions with i connected components in the algorithm PDFO, the relation*

$$\sum_{e \in F_j^t} w(e) \;\leq\; \omega \left[\sum_{\pi \in \Pi} ((|\pi| - j) \cdot y_\pi^t) - \sum_{v \in V} (B_v \cdot \lambda_v^t) \right] \tag{10}$$

holds for each j, $1 \leq j \leq i$.

Proof. For $t = 0$ this is obviously true. Let F_j^t be the spanning forest produced by the algorithm MSF_j in the t-th iteration and let $w^t(F_j^t)$ be the weight of this spanning forest with respect to the weight function w^t. As the weights can only increase during the run of PDFO,

$$w(F_j^t) \leq w^t(F_j^t) = \sum_{e \in F_j^t} w^t(e) = \sum_{\pi \in \Pi} (|\pi| - j) \cdot y_\pi^t$$

holds. Using the invariant given in Lemma 7 we get

$$w(F_j^t) \;\leq\; \omega \left[\sum_{\pi \in \Pi} ((|\pi| - j) \cdot y_\pi^t) - \sum_{v \in V} (B_v \cdot \lambda_v^t) \right]$$

□

Lemma 8 shows that in each iteration the weight of a spanning forest with j, $1 \leq j \leq i$, is only a constant times the weight of an optimal solution. It remains to show an upper bound on the runtime of PDFO. To do this we first consider the time to produce a new solution x^{t+1} from the current solution x^t.

Lemma 9. *The solution x^{t+1} can be computed from x^t in time $O(n^2)$.*

Proof. If the computation of x^{t+1} is carried out in step 3a of the algorithm introducing the lightest edge for the weight function w^{t+1} into x^t that does not create a cycle yields x^{t+1}. This can be done in time $O(n^2)$ by inspecting every edge at most once. In the other case x^t is a minimum spanning forest with i connected components with respect to w^t and x^{t+1} is a minimum spanning forest with i connected components with respect to the updated weight function w^{t+1}. To determine x^{t+1} we have to compute ϵ^t and execute the resulting exchange operation. For the computation of d^t we use an integer array of size n and store at position j, $0 \leq j \leq n-1$, the sum over the B_v-values with vertices of normalized degree j in F_i^t. This can be done in time $O(n)$ using a breath first search traversal on F_i^t in which we compute the B_v value for the current vertex v in the traversal and add the value to the value of the corresponding position in the array. After that we determine the values $\sum_{v \in U_j} B_v$, $0 \leq j \leq n-1$, one after another starting with U_0. Note that $\sum_{v \in U_0} B_v = \sum_{v \in V} B_v$. The value $\sum_{v \in U_{j+1}} B_v$ can be computed by subtracting from $\sum_{v \in U_j} B_v$ the entry at position j in the array. Each computation can be done in constant time based on the corresponding array values. Hence, the value d^t due to Lemma 5 can be determined in time $O(n)$. To compute the ϵ^t value we determine the exchange operation that leads to a primal solution of $\mathrm{MSF}_i(G, w^{t+1})$. Note that ϵ^t is the smallest weight increase such that deleting an edge adjacent to at least one vertex of U_{d^t} and inserting an edge adjacent to vertices that are not contained in U_{d^t-1} yields a minimum spanning forest with i components for the weight function w^{t+1}. Similar to Lemma 4 we consider two possibilities for the exchange operation.

First we investigate the case where introducing an edge e connects two components of the current forest F_i^t. Then another edge from the resulting forest has to be removed to create a solution with i connected components. Introducing the edge e with smallest weight that is not incident to vertices of U_{d^t-1} and deleting the edge $e' \in F_i^t$ that has the largest weight of all edges incident to vertices of U_{d^t} in F_i^t leads to the desired exchange operation with the smallest weight increase. Each edge of G has to be examined once, which gives an upper bound of $O(n^2)$ on the runtime in this case. Let $\epsilon^{t'}$ be the value obtained by this exchange operation.

The other possibility to get a smaller value than $\epsilon^{t'}$ is to introduce into F_i^t an edge that creates a cycle. Then we have to delete one edge of this cycle. We use a depth first search traversal of F_i^t from every vertex $v \in V$. Let w be the current vertex in this traversal. Assume that there is an edge $e = (v, w)$ in E and that v and w are not contained in U_{d^t-1}. Otherwise we can continue the traversal since the pair (v, w) does not fulfill the properties for the exchange operation. Let w_i be the largest weight of an edge e' in the path from v to w that is incident to vertices of U_{d^t}. If no such edge edge e' exists in the path from v to w, e can not participate in the exchange operation we are looking for. The weight increase of introducing e and deleting e' can be computed in constant time, and the w_i variables can be maintained in constant time per step of the traversal using stacks. Hence, we can determine the exchange operation with the smallest weight increase in the second case in time $O(n^2)$. Let $\epsilon^{t''}$ be weight increase of this exchange operation. Choosing $\epsilon^t = \min\{\epsilon^{t'}, \epsilon^{t''}\}$ and computing a primal solution of MSF_i^{t+1} by executing the corresponding exchange operation gives the stated upper bound. In addition we update the weight with respect to w^{t+1} for the next iteration which can be done in time $O(n^2)$. □

Theorem 2. *The algorithm PDFO computes for any $b > 1$ and $\omega > 1$ in time $O(n^{3+2/\ln b})$ a set of solutions that includes for each $i \in \{1, \ldots n\}$ a minimum spanning forest with i connected components with each vertex degree at most $\alpha \cdot B_v + 2\log_b n + 1$ and total weight at most $\omega \cdot w(F_i^*)$, where $\alpha = \max\{\omega/(\omega-1), \omega\}$ and $w(F_i^*)$ is the minimum weight of any spanning forest with i connected components satisfying the degree bounds.*

Proof. As long as the algorithm has not achieved a solution with i connected components such that the vertex degree is at most $\alpha \cdot B_v + 2\log_b n + 1$, the right hand side of (10) is ω times the optimal value of the dual objective function. This implies that the weight of a minimum spanning forest with i connected components for the weight function w^t is at most ω times the value of an optimal solution obeying the degree bounds for each $j \in \{1, \ldots, i\}$. Hence, the solutions introduced into the set S in step 3c of PDFO have the stated approximation quality. As each possible value of i is considered in the run of PDFO, the set S includes after termination for each i, $1 \leq i \leq n$, a solution with i connected components that has the desired approximation quality.

In the following we give an upper bound of $O(n^{3+2/\ln b})$ on the runtime until the algorithm terminates. A new solution x^{t+1} can be computed from the current solution in time $O(n^2)$ due to Lemma 9. It remains to bound the number of primal solutions x^t that have to be computed until the algorithm terminates.

The number of solution that are computed in step 3a of the algorithm is at most $n-1$ as the number of connected components is bound by n. In the inner while-loop we compute for the solution with i connected components new primal solutions as long as we have not reached a solution with i connected components that has the desired approximation quality.

We consider a modification of the potential function g introduced in Lemma 3. Consider a solution s, let $\hat{\Delta}(s)$ be the maximum normalized degree of s, and let $\hat{d}_i$, $0 \leq i \leq n-1$, be the number of vertices with normalized degree i in this solution. The potential of a solution s is given by $h(s) := \sum_{j=0}^{\lceil 2\log_b n \rceil + 1} \hat{d}_{r+j}(s) \cdot e^j$, where $r = \hat{\Delta}(s) - \lceil 2 \log_b n \rceil$. Assume the a minimum spanning forest with i, $2 \leq i \leq n$, has been computed. After initialization this is true for $i = n$. The solution of $\mathrm{MSF}_{i-1}(G, w^t)$ differs from $\mathrm{MSF}_i(G, w^t)$ by one single edge that is additionally introduced into $\mathrm{MSF}_{i-1}(G, w^t)$ at any time t. Introducing this edge into s_i, the potential increases by $O(n^{2/\ln b})$. Each iteration of the inner while-loop reduces the potential by a constant fraction. This implies that the number of improvements until the value of i has been reduced in the run of PDFO is $O(n^{2/\ln b})$. The value of i decrease $n-1$ times. Hence, the total number of iterations of the inner while loop is bounded by $O(n^{1+2/\ln b})$. □

Note that choosing $b = e$, where $e = 2.71\ldots$, the upper bound on the runtime is $O(n^5)$, the degree bound is $O(B_v \log n)$, and the weight of a solutions with i connected components introduced into the set S in step 3c is at most a constant times the weight of any minimum spanning forest with i connected components obeying the degree bounds. Therefore we can state the following corollary.

Corollary 1. *The algorithm PDFO computes for each $i \in \{1, \ldots, n\}$ in time $O(n^5)$ a solution with i connected components in which each vertex v has degree $O(B_v + \log n)$ and whose weight is at most a constant times an optimal solution with i connected components satisfying the degree bounds.*

4 Conclusions

We have given faster approximation algorithms for the generalization of two NP-hard spanning tree problems. These algorithms are based on the multi-objective view which enables use to reduce the number of necessary improvements by a factor n until the approximation guarantee can be given. Based on these observations we have given an algorithm that computes for each $i \in \{1, \ldots, n\}$ a minimum spanning forest with i components and degree at most $b \cdot \Delta_i^* + \lceil \log_b n \rceil$ in a total time of $O(n^{3+1/\ln b})$. In the case of nonuniform degree bounds we have presented an algorithm that runs in time $O(n^5)$ and computes for each $i \in \{1, \ldots, n\}$ a spanning forest in which each vertex has degree $O(B_v + \log n)$ and the weight is a most a constant times the weight of a minimum spanning forest with i components obeying the given degree bounds.

References

1. Camerini, P. M., Galgiati, G, and Maffioli, F. (1980). Complexity of spanning tree problems. European Journal on Operation Research, 5: 346-252.
2. Caro, Y., Krasikov, I., and Roditty, Y. (1991). On the largest tree of a given maximum degree in a connected graph. Journal of Graph Theory, 15: 7-13.
3. Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). Introduction to Algorithms. 2nd Edition, McGraw Hill, New York.
4. Fischer, T. (1993). Optimizing the Degree of Minimum Weight Spanning Trees. Technical Report 93-1338, Department of Computer Science, Cornell University, Ithaca, NY, USA.
5. Fürer, M., and Raghavachari, B. (1990). An NC approximation algorithm for the minimum degree spanning tree problem. In Proc. of the 28th Annual Allerton Conf. on Communication, Control and Computing, 274-281.
6. Fürer, M., and Raghavachari, B. (1992). Approximating the Minimum-Degree Spanning Tree to within One from the Optimal Degree. In Proc. of the third annual ACM-SIAM symposium on Discrete algorithms (SODA), 317-324.
7. Fürer, M., and Raghavachari, B. (1994). Approximating the Minimum-Degree Steiner Tree to within One of Optimal. Journal of Algorithms 17, 409-423.
8. Garey, M. R., Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, New York.
9. Goemans, M. X., and Williamson, D. P. (1995). A general approximation technique for constrained forest problems. SIAM J. Comput. 24, 296-317
10. Könemann, J., and Ravi, R. (2003). Primal-dual meets local search: approximating MST's with nonuniform degree bounds. In Proc, ACM Symposium on Theory of Computing (STOC), 389-395
11. Neumann, F. and Wegener, I. (2005). Minimum spanning trees made easier via multi-objective optimization. Submitted for publication.