Operations Research Letters 45 (2017) 115–118

Contents lists available at ScienceDirect

Operations Research Letters

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

# Approximating bounded-degree spanning trees and connected factors with leaves

Walter Kern, Bodo Manthey [*]

*University of Twente, Department of Applied Mathematics, P. O. Box 217, 7500 AE Enschede, Netherlands*

ARTICLE INFO

*Article history:*
Received 15 September 2016
Received in revised form
12 January 2017
Accepted 12 January 2017
Available online 18 January 2017

*Keywords:*
Approximation algorithms
Bounded-degree spanning trees
Connected graph factors
Network design

ABSTRACT

We present constant factor approximation algorithms for the following two problems: First, given a connected graph $G = (V, E)$ with non-negative edge weights, find a minimum weight spanning tree that respects prescribed upper bounds on the vertex degrees. Second, given prescribed (exact) vertex degrees $d = (d_i)_{i \in V}$, find a minimum weight connected $d$-factor. Constant factor approximation algorithms for these problems were known only for the case that $d_i \geq 2$ for all $i \in V$.

## 1. Introduction

Finding low-cost spanning subgraphs with prescribed degree and connectivity requirements is a fundamental problem in the area of network design. The goal is to find a cheap, connected subgraph that meets the degree constraints. Most variants of such problems are NP-hard. Because of this, finding good approximation algorithms for such network design problems has been the topic of a significant amount of research [2,4–9,12–17].

In this paper, we study the problem of finding low-cost spanning connected subgraphs with degree constraints, where violation of the degree constraint is not allowed. The degree constraints are either upper bounds or have to be met exactly.

Minimum weight subgraphs with prescribed vertex degrees can be found efficiently using Tutte's reduction to the perfect matching problem [18,20]. But asking for connectedness in addition makes the problem NP-hard [3]. For instance, asking for a 2-regular, connected spanning subgraph of minimum weight is the NP-hard traveling salesman problem (TSP) [11, Problem ND22]. Also finding spanning trees with given upper bounds for the degrees of the nodes is NP-hard [10].

Approximation algorithms address three variants of the problem: First, one may relax the degree constraints and compare the weight of the solution computed (subject to the relaxed requirements) with an optimal solution that has to satisfy the requirements strictly [6,19]. Second, one may view the problem as a bicriteria optimization problem, where one objective is the weight and the other objective is the violation of the degree constraints [8,9,15–17]. Third, one may insist on meeting the degree constraints exactly [5,7]. In this paper, we consider the third variant.

A main obstacle seems to be vertices that are required to have degree 1. In fact, existing approximation algorithms [5–7] only work when the minimum degree requirement is at least 2, and it has been raised as an open problem [5,7] to approximate network design problems in the presence of vertices that must have degree 1.

### 1.1. Problem definition

In this paper, we consider three different optimization problems. In each case, an instance consists of a simple undirected complete graph $G = (V, E)$ with edge weights $w$ that satisfy the triangle inequality and given $d = (d_i)_{i \in V}$ to be interpreted as either prescribed vertex degrees or upper bounds thereof. For $F \subseteq E$, let $\deg_F(i)$ be the degree of node $i \in V$ in the graph $(V, F)$. Furthermore, $w(F) = \sum_{e \in F} w(e)$ is the total weight of the edge set $F$. In case of multi-graphs, edges are counted with multiplicities (both for the degree and the total weight).

In the *bounded-degree minimum spanning tree problem* (denoted by BMST), we are to compute a tree $T \subseteq E$ of minimum weight with the additional condition that $\deg_T(i) \leq d_i$ for all $i \in V$. We

[*] Corresponding author.
*E-mail addresses:* w.kern@utwente.nl (W. Kern), b.manthey@utwente.nl (B. Manthey).

http://dx.doi.org/10.1016/j.orl.2017.01.002
0167-6377/© 2017 Elsevier B.V. All rights reserved.

call such a tree a *d-bounded tree*. We denote a minimum weight $d$-bounded tree by $\text{Tree}_d$, breaking ties arbitrarily.

In the *connected factor problem* (denoted by ConnFact), our goal is to compute a connected $d$-factor $F$ of minimum weight. Multiple edges are not allowed. This means that $(V, F)$ must be connected and $\deg_F(i) = d_i$ for all vertices $i \in V$. We denote a minimum weight connected $d$-factor (without multiple edges) by $\text{ConnFact}_d$, again breaking ties arbitrarily.

The *connected factor problem with multiple edges* (denoted by ConnMFact) is similar to ConnFact. The only difference is that multiple edges are allowed. So a solution $F$ is a multi-set of edges such that $(V, F)$ is connected and $\deg_F(i) = d_i$. We denote a minimum weight connected $d$-factor with possibly multiple edges by $\text{ConnMFact}_d$, again breaking ties arbitrarily.

Minimum weight $d$-factors (without connectivity requirement) can be computed in polynomial time with and without multiple edges. We denote a minimum weight $d$-factor without multiple edges by $\text{Fact}_d$ and one with multiple edges allowed by $\text{MFact}_d$.

### 1.2. Previous results

Because connected factor problems generalize the TSP, no polynomial-time constant factor approximation algorithms are possible in general graphs or without the triangle inequality [21, Theorem 2.9]. Therefore, we assume that input graphs are complete and that the edge weights satisfy the triangle inequality.

We restrict ourselves in the discussion of previous results to the case of algorithms that meet the degree requirements exactly. Fukunaga and Nagamochi [7] considered the problem of finding a minimum weight $k$-edge-connected subgraph that meets given degree requirements precisely. They allow multiple edges between vertices (which seem to simplify the problem considerably, because it is possible to add connections between arbitrary vertices, independent of whether the corresponding edge is already present). For this relaxed variant of the problem, they obtain approximation ratios of 2.5 for even $k$ and $2.5 + \frac{1.5}{k}$ for odd $k$ if the minimum degree requirement is at least 2. For the case of simple connectivity, Cornelissen et al. [5] devised an approximation algorithm with ratio 3. Fekete et al. [6] devised an approximation algorithm for the bounded-degree spanning tree problem that achieves an approximation ratio of roughly 2.

### 1.3. Our contribution

All three algorithms mentioned in the previous section require that all prescribed $d_i$ are at least 2, and Fukunaga and Nagamochi [7] and Cornelissen et al. [5] raised the question if constant factor approximation algorithms also exist in case some of the $d_i$ are equal to 1. We give an affirmative answer to this question.

First, we present a factor 3-approximation algorithm for BMST (Section 2). Then we use this algorithm to get factor 7 approximation algorithms for both ConnFact and ConnMFact (Section 3).

The approximation ratios that we achieve are considerably worse than the ratios of roughly 2 [6], 3 [5], 4 [7] for BMST, ConnFact, and ConnMFact, respectively, that hold if we forbid degree 1 nodes. (The 4-approximation for ConnMFact without degree-1-nodes can easily be improved to a factor 3-approximation by adapting the algorithm by Cornelissen et al. [5].) The obvious open question is whether this gap can be closed.

## 2. Bounded-degree spanning trees

We start with a simple observation, based on the standard construction of Hamilton paths by doubling a minimum spanning tree.

**Lemma 1.** *Given an undirected, complete graph $G$ with edge weights $w$ that satisfy the triangle inequality and an edge $e_0 = \{i_0, j_0\} \in E$, we can compute in polynomial time a Hamiltonian path $P$ with endpoints $i_0$ and $j_0$ such that $w(P) \le 2w(T)$, where $T \subseteq E$ is a spanning tree that contains $e_0$ and has minimum weight among all such trees.*

**Proof.** After inserting the edge $e_0$ first, we connect all nodes in a Kruskal-like manner. This yields a spanning tree $T$ containing $e_0$ that has minimum weight among all such trees. We duplicate all edges of $T$ to obtain a Eulerian graph $T'$. Then we traverse $T'$, starting with the edge $e_0$ and taking shortcuts to obtain a Hamiltonian cycle $H$. By construction, $w(H) \le 2w(T)$. We obtain $P$ by removing $e_0$ from $H$. □

In what follows, we often distinguish between nodes with prescribed degree $d_i = 1$ and other nodes. For this reason, we define $V_{=1} = \{i \in V \mid d_i = 1\}$ and $V_{\ge 2} = \{i \in V \mid d_i \ge 2\}$. Any $d$-bounded tree $T$ consists of an *interior tree* $T_{\text{int}}$ that connects only the $V_{\ge 2}$ nodes and to which the $V_{=1}$ nodes are attached. We may assume that $T_{\text{int}}$ connects at least two nodes. Otherwise, $|V_{\ge 2}| \le 1$ and the problem becomes trivial. The most challenging part is to determine how the vertices in $V_{=1}$ are attached to the interior tree.

To address this problem, we proceed in two steps. In the first step, we compute a forest that spans all of $V_{=1}$ and a subset of $V_{\ge 2}$ without violating the degree constraints. This forest is computed by solving an appropriate minimum-cost flow problem. In the second step, we connect the components of this forest along a Hamiltonian path through a subset of the $V_{\ge 2}$ nodes. In this way, we construct a tree whose leaves are a subset of $V_{=1}$. Note that an optimal tree can also have leaves from $V_{\ge 2}$.

Let us describe the first step. In what follows, we assume that we know an edge $e_0 = \{i_0, j_0\} \in \text{Tree}_d$ in the interior tree of the unknown optimum solution $\text{Tree}_d$. (In our algorithm, we fix $i_0 \in V_{\ge 2}$ arbitrarily, try all possible choices of $j_0 \in V_{\ge 2} \setminus \{i_0\}$, and take the best outcome.) Removing $e_0$ splits the unknown tree $\text{Tree}_d$ into two subtrees. To outline the intuition behind our approach, consider $i_0$ and $j_0$ as the roots of these subtrees, and direct all edges in these two subtrees towards $i_0$ and $j_0$, respectively. We may interpret the subtrees as "flows" from the $V_{=1}$ nodes towards the roots $i_0$ and $j_0$, respectively. In this sense, the two subtrees define a solution to the flow problem (with node capacities) described below.

Consider the following flow problem $\text{MCF}_{e_0}$: The underlying graph has vertex set $V \cup \{r\}$, where $r \notin V$ is a new node, and edge set $(E \setminus \{e_0\}) \cup \{\{i, r\} \mid i \in V_{\ge 2}\}$. All edges $e \in E \setminus \{e_0\}$ have a capacity of 1 in both directions and costs of $w(e)$ per unit of flow. Each node $i \in V_{\ge 2}$ has a node capacity of $d_i - 1$ (this means that at most $d_i - 1$ units of flow may pass through $i$). The edges $\{i, r\}$ for $i \in V_{\ge 2}$ are *overflow edges*. They have cost 0. For $i \in V_{\ge 2} \setminus \{i_0, j_0\}$, edge $\{i, r\}$ has a capacity of $d_i - 2$. For $i \in \{i_0, j_0\}$, edge $\{i, r\}$ has a capacity of $d_i - 1$. The task is to find a minimum-cost flow from the $V_{=1}$ nodes, each having a supply of 1, to the new root node $r$, which has a demand of $|V_{=1}|$. Such a minimum-cost flow can be computed in polynomial time [1].

The set $\text{Tree}_d \setminus \{e_0\}$ defines a solution $f_{\text{Tree}}$ of this flow problem as follows: Recall that we direct all edges in the two subtrees of $\text{Tree}_d \setminus \{e_0\}$ towards their roots $i_0$ or $j_0$, respectively. On every arc $e = \{i, j\}$ in the directed tree $\text{Tree}_d \setminus \{e_0\}$, we have a flow of 1 (towards $i_0$ or $j_0$). Thus, in particular, each $i \in V_{=1}$ has an outflow of 1. If a node $i \in V_{\ge 2} \setminus \{i_0, j_0\}$ has degree $\ell$ ($2 \le \ell \le d_i$) in $\text{Tree}_d$, then in the directed tree, it has $\ell - 1$ incoming arcs and one outgoing arc (in direction to the root $i_0$ or $j_0$). Thus its total inflow equals $\ell - 1$ and we send $\ell - 2$ units of outflow directly to $r$ on the overflow arc from $i$ to $r$. Note that the node capacity constraint (throughput at most $d_i - 1$) is met. If $i \in \{i_0, j_0\}$ has degree $\ell$ ($2 \le \ell \le d_i - 1$) in $\text{Tree}_d$, then its inflow equals $\ell$ units, which we route to $r$ on the overflow arc $\{i, r\}$. This, again, also respects the node capacity constraints.