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

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 $i \in V$.

Keywords: approximation algorithms, bounded-degree spanning trees, connected graph factors, network design

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, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 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].

*Corresponding author

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, 16, 17]. Third, one may insist on meeting the degree constraints exactly [5, 7]. In this paper, we consider the third variant.

A main obstacle seem to be vertices that are required to have degree 1. In fact, existing approximation algorithms [5, 6, 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 call such a tree a *d-bounded tree*. We denote a minimum weight d-bounded tree by 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 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 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 Fact_d and one with multiple edges allowed by 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 with-

out 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 seems 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 $w(P) \leq 2w(T)$, 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, $w(H) \leq 2w(T)$. 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 $V_{\geq 2} = \{i \in V \mid d_i \geq 2\}$. Any d-bounded tree T consists of an *interior tree* T_{int} that connects only the $V_{\geq 2}$ nodes and to which the $V_{=1}$ nodes are attached. We may assume that T_{int} connects at least two nodes. Otherwise, $|V_{\geq 2}| \leq 1$ 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 $V_{\geq 2}$ 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 $V_{\geq 2}$ 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 $V_{\geq 2}$.

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 Tree_d. (In our algorithm, we fix $i_0 \in V_{\geq 2}$ arbitrarily, try all possible choices of $j_0 \in V_{\geq 2} \setminus \{i_0\}$, and take the best outcome.) Removing e_0 splits the unknown tree 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 MCF_{e_0}: The underlying graph has vertex set $V \cup \{r\}$, where $r \notin V$ is a new node, and edge set $(E \setminus \{e_0\}) \cup \{\{i, r\} \mid i \in V_{\geq 2}\}$. 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 $i \in V_{\geq 2}$ 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 $i \in V_{\geq 2}$ are *overflow edges*. They have cost 0. For $i \in V_{\geq 2} \setminus \{i_0, j_0\}$, 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_{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 $i \in V_{\geq 2} \setminus \{i_0, j_0\}$ has degree ℓ ($2 \leq \ell \leq d_i$) in 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 ℓ ($2 \leq \ell \leq d_i - 1$) in Tree_d, then its inflow equals ℓ units, which we route to r on the overflow arc $\{i, r\}$. This, again, also respects the node capacity constraints.

The cost of f_{Tree} is equal to $w(\text{Tree}_d \setminus \{e_0\}) = w(\text{Tree}_d) - w(e_0)$.

Now let $f^\star$ be any integral optimal solution of MCF_{e_0}. We define the *support* of $f^\star$ as the set $S^\star \subseteq E \setminus e_0$ of edges that carry positive flow in either direction. An integral optimal solution of minimum support can be computed efficiently by adding $\varepsilon < \frac{w_{\min}}{|E|}$ to the costs of all edges in $E \setminus \{e_0\}$ in the flow problem, where $w_{\min}$ is the smallest positive edge weight. If S is the support of an integral solution f, then the cost of f is equal to $w(S)$.

Lemma 2. *Let $f^\star$ be an integral optimum solution of MCF_{e_0} with minimum support $S^\star$. Then we have the following properties:*

1. $w(S^\star) \leq w(\text{Tree}_d)$.
2. *$S^\star$ is a forest.*
3. $\deg_{S^\star}(i_0) \leq d_{i_0} - 1$ *and* $\deg_{S^\star}(j_0) \leq d_{j_0} - 1$.
4. *Each connected component of $S^\star$ contains i_0 or j_0 or a node $i \in V_{\geq 2}$ with $\deg_{S^\star}(i) \leq d_i - 2$. (We call such a node i a* root.*)*

Proof. (1): The set $\text{Tree}_d \setminus \{e_0\}$ of edges defines a solution of MCF_{e_0} as described above.

(2): If any $i \in V_{\geq 2}$ has at least two outgoing arcs (corresponding to out-flow in $f^\star$), say, $\{i, j\}$ and $\{i, k\}$ for $j, k \in V_{\geq 2}$, then we could reroute one unit of flow, say from i to r via k to the overflow edge $\{i, r\}$. This does not increase the cost of the flow. (Note that as long as the out-flow of i to other nodes in $V_{\geq 2}$ is at least 2, the overflow edge is not saturated because of the node capacity of i.) This rerouting would decrease the support size, which contradicts the assumption that $f^\star$ has minimum support. We conclude that in $f^\star$, any $V_{\geq 2}$ nodes (and also any $V_{=1}$ node) has at most one outgoing arc towards $V_{\geq 2}$ nodes carrying flow. Hence, if $S^\star$ contains cycles, then these correspond to directed cycles of flow 1 arcs. Removing such a cycle would decrease the support size, again yielding a contradiction.

(3): Node i_0 has an in-flow of at most $d_{i_0} - 1$ and out-flow only towards r. The same holds for j_0.

(4): Any component that contains neither i_0 nor j_0 must contain a root $i \in V_{\geq 2}$ that has no out-flow towards another $V_{\geq 2}$ node. Thus, i gets rid of all its inflow via the corresponding overflow edge $\{i, r\}$. This has capacity $d_i - 2$, so the number of in-flow arcs, which is equal to $\deg_{S^\star}(i)$, can be at most $d_i - 2$. □

Now we are almost done. Given $S^\star$, the support of a flow as in Lemma 2, we connect the connected components via a Hamilton path P with endpoints i_0 and j_0 as in Lemma 1: In each component of $S^\star$ that contains neither i_0 nor j_0, we pick a root i of degree at most $d_i - 2$ in $S^\star$. Such a root exists by Lemma 2. Then we connect the components of $S^\star$ by following P, starting in i_0, ending in j_0 and skipping all other vertices except the root nodes chosen. This yields a d-bounded tree T of weight $w(T) \leq w(S^\star) + w(P) \leq w(\text{Tree}_d) + 2w(\text{Tree}_d) \leq 3w(\text{Tree}_d)$. Algorithm 1 summarizes this procedure, and we obtain the following theorem.

Algorithm 1: A 3-approximation for BMST.

input : undirected, complete graph $G = (V, E)$, edge weights w satisfying the triangle inequality, degrees $(d_i)_{i \in V}$
output: d-bounded tree T

select an arbitrary $i_0 \in V_{\geq 2}$
for $j_0 \in V_{\geq 2} \setminus \{i_0\}$ **do**
  let $e_0 = \{i_0, j_0\}$
  compute an optimal solution $f^\star$ of minimum support of MCF_{e_0}
  extract $S^\star$ from $f^\star$ as described
  compute a Hamiltonian path P with endpoints i_0 and j_0
  restrict P to the "roots" of the connected components of $S^\star$ by taking shortcuts
  $T_{e_0} \leftarrow P \cup S^\star$
end
let T be the lightest tree among all T_{e_0}

Theorem 3. *Algorithm 1 is a polynomial-time 3-approximation for* BMST.

3. Connected Factors

The idea to approximate connected factors is as follows: we compute a minimum-weight d-factor F, which is not necessarily connected, and a d-bounded tree T. As long as the d-factor F is not connected, there exists an edge $e \in T \setminus F$ that we can add. In order to maintain the degrees, we remove one edge of each endpoint of e and add another edge. The following lemma states that this always works. In particular, it is crucial that we never remove edges of the d-bounded tree.

Lemma 4. *Let T be a d-bounded tree, and let F be a d-factor. If F is not connected, then we can find an edge $e = \{i, j\} \in T \setminus F$ and vertices i' and j' with the following properties:*

1. *e connects two components of F.*
2. *$\{i, i'\}, \{j, j'\} \in F \setminus T$.*
3. *$\{i', j'\} \in E \setminus F$.*

Proof. Since T is connected and F is not connected, there exist an edge $e = \{i, j\} \in T \setminus F$. We have $\deg_T(i) \leq d_i = \deg_F(i)$ and $\deg_T(j) \leq d_j = \deg_F(j)$. Since $e \in T \setminus F$, there must be edges $\{i, i'\}, \{j, j'\} \in F \setminus T$. Since i and j are in different components, we have $\{i', j'\} \notin F$. □

Theorem 5. *Algorithm 2 is a polynomial-time 7-approximation for* ConnFact.

Proof. Let i, j, i', j' be four nodes chosen by Algorithm 2 in some iteration. By the triangle inequality, we have $w(i', j') \leq w(i', i) + w(i, j) + w(j, j')$. Hence,

Algorithm 2: A 7-approximation for ConnFact.

input : undirected complete graph $G = (V, E)$, edge weights w satisfying the triangle inequality, degrees $(d_v)_{v \in V}$
output: connected d-factor $\tilde{F}$ (without multiple edges)

compute a 3-approximation T of a d-bounded spanning tree using Algorithm 1
$F \leftarrow \mathrm{Fact}_d$
while F *is not connected* **do**
  choose an edge $e = \{i, j\} \in T \setminus F$ that connects two different components of F
  choose i', j' as in Lemma 4
  $F \leftarrow \big(F \cup \big\{\{i, j\}, \{i', j'\}\big\}\big) \setminus \big\{\{i, i'\}, \{j, j'\}\big\}$
end
$\tilde{F} \leftarrow F$

the weight of F increases by at most $2w(i, j)$ in this iteration. Since no edge $\{i, j\}$ of T is considered more than once, we have $w(\tilde{F}) \leq w(\mathrm{Fact}_d) + 2w(T) \leq w(\mathrm{Fact}_d) + 6w(\mathrm{Tree}_d) \leq 7w(\mathrm{ConnFact}_d)$, where $\tilde{F}$ is the connected d-factor output by the algorithm.

By construction, F is a d-factor initially, and it remains a d-factor throughout the execution of the algorithm. Furthermore, F is connected since, by Lemma 4 and the construction of the algorithm, any edge of T that is added by the algorithm is never removed later on. Therefore, after less than n iterations of the while loop, F must be connected. □

The algorithm above also works for the case of ConnMFact, where multiple edges are allowed. We just have to replace the initialization of F by MFact_d. The analysis of the approximation ratio follows from $w(\mathrm{MFact}_d) \leq w(\mathrm{ConnMFact}_d)$ and $w(\mathrm{Tree}_d) \leq w(\mathrm{ConnMFact}_d)$. Hence, we get the following result.

Corollary 6. *There exists a polynomial-time 7-approximation for* ConnMFact.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, 1993.

[2] Yuk Hei Chan, Wai Shing Fung, Lap Chi Lau, and Chun Kong Yung. Degree bounded network design with metric costs. *SIAM Journal on Computing*, 40(4):953–980, 2011.

[3] F. Cheah and Derek G. Corneil. The complexity of regular subgraph recognition. *Discrete Applied Mathematics*, 27(1-2):59–68, 1990.

[4] Joseph Cheriyan and Adrian Vetta. Approximation algorithms for network design with metric costs. *SIAM Journal on Discrete Mathematics*, 21(3):612–636, 2007.

[5] Kamiel Cornelissen, Ruben Hoeksma, Bodo Manthey, N. S. Narayanaswamy, C. S. Rahul, and Marten Waanders. Approximation algorithms for connected graph factors of minimum weight. *Theory of Computing Systems*, to appear.

[6] Sándor P. Fekete, Samir Khuller, Monika Klemmstein, Balaji Raghavachari, and Neal E. Young. A network-flow technique for finding low-weight bounded-degree spanning trees. *Journal of Algorithms*, 24(2):310–324, 1997.

[7] Takuro Fukunaga and Hiroshi Nagamochi. Network design with edge-connectivity and degree constraints. *Theory of Computing Systems*, 45(3):512–532, 2009.

[8] Takuro Fukunaga and Hiroshi Nagamochi. Network design with weighted degree constraints. *Discrete Optimization*, 7(4):246–255, 2010.

[9] Takuro Fukunaga and R. Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. In *Proc. 53rd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 263–272. IEEE Computer Society, 2012.

[10] Martin Fürer. Degree-bounded trees. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 231–233. Springer, 2008.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[12] Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *Journal of Computer and System Sciences*, 79(5):725–736, 2013.

[13] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *Journal of the ACM*, 41(2):214–235, 1994.

[14] Guy Kortsarz and Zeev Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.

[15] Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009.

[16] Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM Journal on Computing*, 42(6):2217–2242, 2013.

[17] Lap Chi Lau and Hong Zhou. A unified algorithm for degree bounded survivable network design. In Jon Lee and Jens Vygen, editors, *Proc. 17th Int. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, volume 8494 of *Lecture Notes in Computer Science*, pages 369–380. Springer, 2014.

[18] László Lovász and Michael D. Plummer. *Matching Theory*, volume 121 of *North-Holland Mathematics Studies*. Elsevier, 1986.

[19] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *Journal of the ACM*, 62(1):1:1–1:19, 2015.

[20] William T. Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6:347–352, 1954.

[21] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.