On Variants of the Spanning Star Forest Problem*

Jing He and Hongyu Liang

Institute for Interdisciplinary Information Sciences,
Tsinghua University, Beijing, China
{he-j08,lianghy08}@mails.tsinghua.edu.cn

Abstract. A star forest is a collection of vertex-disjoint trees of depth at most 1, and its size is the number of leaves in all its components. A spanning star forest of a given graph G is a spanning subgraph of G that is also a star forest. The *spanning star forest problem* (SSF for short) [14] is to find the maximum-size spanning star forest of a given graph. In this paper, we study several variants of SSF, obtaining first or improved approximation and hardness results in all settings as shown below.

1. We study SSF on graphs of minimum degree $\delta(n)$, n being the number of vertices in the input graph. Call this problem $(\geq \delta(n))$-SSF. We give an (almost) complete characterization of the complexity of $(\geq \delta(n))$-SSF with respect to $\delta(n)$ as follows.
   - When $1 \leq \delta(n) \leq O(1)$, $(\geq \delta(n))$-SSF is APX-complete.
   - When $\omega(1) \leq \delta(n) \leq O(n^{1-\epsilon})$ for some constant $\epsilon > 0$, $(\geq \delta(n))$-SSF is NP-hard but admits a PTAS.
   - When $\delta(n) \geq \omega(n^{1-\epsilon})$ for every constant $\epsilon > 0$, $(\geq \delta(n))$-SSF is not NP-hard assuming Exponential Time Hypothesis (ETH).
2. We investigate the spanning k-tree forest problem, which is a natural generalization of SSF. We obtain the first inapproximability bound of $1 - \Omega(\frac{1}{k})$ for this problem, which asymptotically matches the known approximation ratio of $1 - \frac{1}{k+1}$ given in [13]. We then propose an approximation algorithm for it with a slightly improved approximation ratio of $1 - \frac{1}{k+2}$.
3. We prove that SSF cannot be approximated to any factor larger than $\frac{244}{245}$ in polynomial time, unless $P = NP$. This improves the previously best known bound of $\frac{259}{260}$ [14].

1 Introduction

All graphs considered in this paper are undirected and simple, that is, they contain neither self-loops nor parallel edges. A *star* is a tree of depth at most 1, and its root is called the *center* of it. A star forest is a collection of vertex-disjoint

* This work was supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174. Part of this work was done while the authors were visiting Cornell University.

M. Atallah, X.-Y. Li, and B. Zhu (Eds.): FAW-AAIM 2011, LNCS 6681, pp. 70–81, 2011.

stars, and its size is the number of leaves in all its components. A *spanning star forest* of a graph G is a spanning subgraph of G that is also a star forest. The *spanning star forest problem* (SSF for short) [14] is the problem of finding a maximum-size spanning star forest of a given graph G . This problem has found applications in aligning multiple genomic sequences [14], comparison of phylogenetic trees [5], and the diversity problem in the automobile industry [1].

Spanning star forests are closely related to dominating sets, which are fundamental in graph theory. A dominating set of a graph G is a subset of vertices with the property that every other vertex is adjacent to at least one vertex in it. The *minimum dominating set problem* is to find a smallest dominating set of a given graph. Let G be a graph of order n and F be a spanning star forest of it. Then the collection of all centers of F form a dominating set of G, whose size is precisely n minus the size of F. Conversely, any dominating set D of G naturally induces a spanning star forest of G whose centers are exactly those vertices in D. Thus, the spanning star forest problem and the minimum dominating set problem are equivalent in finding the optimum solution. Nevertheless, they appear totally different in terms of approximation. It is proved by Feige [10] that, unless $NP \subseteq DTIME(n^{O(\log \log n)})$, there is no $(1-\epsilon)\ln n$ approximation algorithm for the minimum dominating set problem. In contrast, a 0.5-approximation to SSF can be easily attained by dividing a spanning tree into alternating levels and choosing odd or even levels to be centers depending on which part induces a larger star forest [14].

In their paper initializing the study of SSF, Nguyen et al. [14] proposed a 0.6-approximation algorithm based on a graph-theoretic observation that every graph of order n has a dominating set of size at most $\frac{2}{5}n$. They also proved that this problem is NP-hard to approximate within a factor of $\frac{259}{260}$. In addition, they introduced the edge-weighted version of SSF, whose goal is to find a spanning star forest of an edge-weighted graph such that the total weight of edges in the star forest is maximized, and showed a 0.5-approximation algorithm for this variant (with the aforementioned idea of alternating levels). The approximation ratio for unweighted SSF was improved to 0.71 by Chen et al. [7] based on a natural linear programming relaxation combined with randomized rounding. Using similar ideas, they gave a 0.64-approximation algorithm for the node-weighted SSF, in which the input graph is node-weighted and the objective is to find a spanning star forest with maximum total weight of leaves. Note that node-weighted SSF is just the complement of the weighted dominating set problem, whose goal is to find a minimum-weight dominating set of the given node-weighted graph. Chakrabarty and Goelin [6] proved that edge-weighted and node-weighted SSF cannot be approximated to $\frac{10}{11}+\epsilon$ and $\frac{13}{14}+\epsilon$ respectively, unless $P = NP$. Athanassopoulos et al. [4] designed a 0.804-approximation for SSF using the idea of semi-local search for k-set cover [9]. He and Liang studied SSF on c-dense graphs [11], where a graph of n vertices is called c-dense if it contains at least $cn^2/2$ edges (see [3]). They showed that, for every fixed $c \in (0, 1)$, SSF on c-dense graphs is APX-hard and can be efficiently approximated within factor $0.804 + 0.196\sqrt{c}$.

1.1 Our Contributions

In this paper, we study several variants of the spanning star forest problem from both algorithmic and hardness points of view, and obtain first or improved approximation and inapproximability results in all settings.

Part 1. We study the spanning star forest problem in graphs with minimum degree constraints. More specifically, we consider the spanning star forest problem on graphs in which every vertex has degree at least $\delta(n)$ for some function $\delta : \mathbb{N} \to \mathbb{N}$, where n is the order (i.e., the number of vertices) of the graph. Denote this problem by $(\geq \delta(n))$-SSF. We prove threshold behaviors for the complexity of the problem with respect to $\delta(n)$. To be precise, we show that:

- When $1 \leq \delta(n) \leq O(1)$, $(\geq \delta(n))$-SSF is APX-complete.
- When $\omega(1) \leq \delta(n) \leq O(n^{1-\epsilon})$ for some constant $\epsilon > 0$, $(\geq \delta(n))$-SSF is NP-hard but allows a PTAS.
- When $\delta(n) \geq \omega(n^{1-\epsilon})$ for every constant $\epsilon > 0$, $(\geq \delta(n))$-SSF is not NP-hard assuming Exponential Time Hypothesis (ETH) [12].

We note that graphs of minimum degree cn for some constant $c > 0$ are called *everywhere-c-dense* graphs [3], which form a subclass of c-dense graphs. Our results indicate that spanning star forest on everywhere-c-dense graphs are easier than on c-dense graphs, since SSF on c-dense graphs is APX-hard for every fixed $c \in (0, 1)$ [11].

Part 2. We next investigate a generalization of SSF, called the spanning k-tree forest problem, which is introduced in [13]. A forest is called a *k-tree forest* if every component of it can be regarded as a tree of depth at most k. The *size* of a k-tree forest is the number of non-root vertices of it (which equals to the number of vertices of the forest minus the number of its components). A *spanning k-tree forest* of G is a spanning subgraph of G that is also a k-tree forest. The goal of the *spanning k-tree forest problem* is to find a maximum-size spanning k-tree forest of the input graph. When $k = 1$, this is just the ordinary spanning star forest problem. We propose a $(1 - \frac{1}{k+2})$-approximation algorithm for the spanning k-tree forest problem, which slightly improves the previously known bound of $1 - \frac{1}{k+1}$ [13]. What is more interesting is that we also prove an asymptotically matching hardness result. Namely, it is NP-hard to approximate the problem to $1 - \Omega(\frac{1}{k})$. To our knowledge, this is the first inapproximability result for this problem.

Part 3. We finally give a short note on the hardness aspect of the original (unweighted) spanning star forest problem. We show, by using the results from Chlebík and Chlebíková [8], that the spanning star forest problem is NP-hard to approximate within ratio $\frac{244}{245}$ even on graphs of maximum degree 5. This improves the previously best known hardness factor of $\frac{259}{260}$ [14].

1.2 Notation Used for Approximation Algorithms

Let Π be a maximization (resp. minimization) problem and $0 < \beta \leq 1$ (resp. $\beta \geq 1$). Let $\mathscr{A}$ be an algorithm for solving Π. We say $\mathscr{A}$ is a *β-approximation algorithm* for Π if, on every instance of Π, it runs in polynomial time and finds a solution with objective value at least (resp. at most) β times the value of the optimum solution. The parameter β is called the *approximation ratio*, or alternatively *approximation factor* or *performance guarantee* of $\mathscr{A}$ for the problem Π. Usually β is considered to be a constant, but sometimes it can also be regarded as a function of the input size or some given parameters. We say Π has a *polynomial time approximation scheme* (PTAS) if for every constant $\epsilon > 0$, there is a $(1-\epsilon)$ (resp. $(1+\epsilon)$)-approximation algorithm for Π. We say Π is *APX-hard* if it does not have a PTAS, and say Π is *APX-complete* if it has a constant factor approximation algorithm and is APX-hard. We refer the readers to [15] for standard definitions and notations not given here.

2 Spanning Star Forest in Graphs with Minimum Degree Constraints

In this section, we study the spanning star forest problem on graphs with minimum degree $\delta(n)$, where n is the order of the graph and $\delta : \mathbb{N} \to \mathbb{N}$ is a function satisfying $1 \leq \delta(n) \leq n-1$ for all $n \in \mathbb{N}$. Denote this restricted problem by $(\geq \delta(n))$-*SSF*. Putting on the requirement $1 \leq \delta(n) \leq n-1$ does not lose generality, since we only consider simple connected graphs. (For disconnected graphs, we simply run the algorithm for connected graphs on every component of it, and gather the obtained star forests to obtain the final solution. This process clearly preserves the approximation ratio.)

By Theorem 1.2.2 in Chap. 1 of [2], it is easy to design a $(1 - O(\frac{\log k}{k}))$-approximation algorithm for $(\geq k)$-SSF. (We temporarily write k instead of $\delta(n)$ for notational simplicity.) The next lemma generalizes the argument to node-weighted graphs.

Lemma 1. *Given any node-weighted graph G of minimum degree k, we can find in polynomial time a dominating set of G of weight at most $\Theta\left(\frac{\log k}{k}\right) \cdot W$, where W is the total weight of all vertices in G.*

Proof. Assume $G = (V, E)$ and $w : V \to \mathbb{R}^+ \cup \{0\}$ be the weight function. Let $p = 1 - \left(\frac{1}{k+1}\right)^{\frac{1}{k}} \in (0, 1)$. We pick every vertex $v \in V$ randomly and independently with probability p, and denote by U_1 the set of picked vertices. Let U_2 be the set of vertices not dominated by U_1, i.e., $U_2 = \{v \in V \mid N(v) \cup \{v\} \subseteq V \setminus U_1\}$, where $N(u)$ denotes the set of neighbors of vertex u. Clearly $U_1 \cup U_2$ is a dominating set of G. Let us bound the expected weight of this set. For all $v \in V$, we have

$$\text{Prob}(v \in U_1) = p$$

and

$$\text{Prob}(v \in U_2) = (1-p)^{1+|N(v)|}.$$

By linearity of expectation,

$$\mathbf{E}(w(U_1 \cup U_2)) = \sum_{v \in V} w(v) \left(p + (1-p)^{1+|N(v)|}\right) \leq \left(p + (1-p)^{1+k}\right) \sum_{v \in V} w(v).$$

Substituting $p = 1 - \left(\frac{1}{k+1}\right)^{\frac{1}{k}}$, we get

$$\mathbf{E}(w(U_1 \cup U_2)) \leq \left(1 - \frac{k}{(k+1)^{1+\frac{1}{k}}}\right) W = \Theta\left(\frac{\log k}{k}\right) \cdot W.$$

Therefore, there exists a dominating set of G of weight at most this much. Note that, for any subset $V' \subseteq V$, if all vertice in V' have been determined whether to belong to U_1 or not, we can calculate the conditional expectation of $w(U_1 \cup U_2)$ in a similar manner. Thus we can find a dominating set of at least this size in deterministic polynomial time by the standard method of conditional expectation. □

Corollary 1. *Every node-weighted graph G with minimum degree k has a spanning star forest of weight at least $\left(1 - \Theta\left(\frac{\log k}{k}\right)\right) W$, where W is the total weight of all vertices in G. Moreover, such a star forest can be found in polynomial time.*

The next claim follows from the fact that the maximum weight of a spanning star forest of G never exceeds the total weight of all vertices in G.

Corollary 2. *There is a $\left(1 - \Theta\left(\frac{\log k}{k}\right)\right)$-approximation algorithm for the node-weighted $(\geq k)$-SSF problem.*

The following theorem is straightforward from the hardness result for the spanning k-tree forest problem (Theorem 5) which will be shown later, and thus we omit its proof.

Theorem 1. *There is a constant $c > 0$ such that, for every fixed integer $k \geq 2$, it is NP-hard to approximate $(\geq k)$-SSF within a factor of $1 - \frac{c}{k}$.*

Notice that $(\geq k)$-SSF is a subproblem of $(\geq k')$-SSF whenever $k \geq k'$. Thus, combined with the fact that SSF has an $\Omega(1)$-approximation algorithm, Theorem 1 directly yields the following corollary, which is the first part of our trichotomy characterization.

Corollary 3. *$(\geq \delta(n))$-SSF is APX-complete if $\delta(n) = O(1)$.*

Now suppose $\delta(n) = \omega(1)$ and $G = (V, E)$ is an input graph to $(\geq \delta(n))$-SSF. By Corollary 2 we can find a $\left(1 - \Theta\left(\frac{\ln(\delta(n))}{\delta(n)}\right)\right)$-approximation algorithm for $(\geq \delta(n))$-SSF. It is easy to check that the approximation factor becomes $1 - o(1)$ when $\delta(n) = \omega(1)$, which immediately implies the following:

Corollary 4. *For any function $\delta(n)$ with $\omega(1) \leq \delta(n) \leq n-1$, $(\geq \delta(n))$-SSF admits a polynomial time approximation scheme (PTAS).*

A PTAS becomes (almost) useless if the problem can be solved optimally in polynomial time. We next show that, for a large range of $\delta(n)$, the restricted problem remains NP-hard, thus establishing the second part of our results.

Theorem 2. *For any fixed $\epsilon > 0$, $(\geq n^{1-\epsilon})$-SSF is NP-hard.*

Proof. Fix ϵ such that $0 < \epsilon < 1$ (the case $\epsilon \geq 1$ is trivial). Let $C = \lceil 1/\epsilon \rceil - 1$. We now describe a polynomial-time reduction from the original (unweighted) SSF problem (or, (≥ 1)-SSF) to unweighted $(\geq n^{1-\epsilon})$-SSF, thus proving the NP-hardness of the latter.

Let $G = (V, E)$ be a simple connected graph. Suppose $V = \{v_1, v_2, \ldots, v_n\}$, where $n = |V|$. We construct a graph $G' = (V', E')$ from G as follows: Let $V' = V \cup \{w_{i,j} : 1 \leq i \leq n, 1 \leq j \leq n^C - 1\}$, and $E' = E \cup \{(v_{i_1}, w_{i_2,j}) : i_1 = i_2$ or $(v_{i_1}, v_{i_2}) \in E, 1 \leq j \leq n^C - 1\} \cup \{(w_{i,j_1}, w_{i,j_2}) : 1 \leq i \leq n, 1 \leq j_1 < j_2 \leq n^C - 1\}$. Note that $|V'| = n + n(n^C - 1) = n^{C+1}$. Let us examine the minimum degree of G'. For every vertex v_i, it is connected to at least one other vertex v'_i with $i' \neq i$, together with $n^C - 1$ nodes $w_{i,j}, 1 \leq j \leq n^C - 1$. Thus its degree is at least n^C. For each vertex $w_{i,j}$, it is adjacent to v_i and $n^C - 2$ other vertices $w_{i,j'}$, $j' \neq j$. Furthermore, since $(v_i, v_{i'}) \in E$ for at least one vertex $v_{i'}$ with $i' \neq i$, $w_{i,j}$ is also adjacent to this $v_{i'}$ according to our definition of E'. Hence $w_{i,j}$ also has degree at least n^C. As $n^C = |V'|^{\frac{C}{C+1}} \geq |V'|^{1-\epsilon}$, the graph G' is a valid input to $(\geq n^{1-\epsilon})$-SSF. Also note that this construction can be finished in polynomial time, since C is a constant.

Let OPT (resp. OPT') denote the size of the maximum spanning star forest of G (resp. G'). We now prove that $OPT' = OPT + n(n^C - 1)$, which will finish the reduction. Let S be the set of centers in the maximum-size spanning star forest of G. We know that S is a dominating set of G, and $OPT = n - |S|$. By the construction of G', the same vertex set S is also a dominating set of G'. Thus we have $OPT' \geq |V'| - |S| = n^{C+1} - |S| = OPT + n(n^C - 1)$.

Now consider the other direction. Let S be the collection of centers in the optimal spanning star forest of G'. If there exists $w_{i,j} \in S$ for some i, j, we construct a new set of centers S' by removing $w_{i,j}$ from S and then declaring v_i as a new center (if $v_i \notin S$); that is, we set $S' = (S \setminus \{w_{i,j}\}) \cup \{v_i\}$. Since every neighbor of $w_{i,j}$ other than v_i is also a neighbor of v_i, S' is a dominating set of G' of cardinality no more than that of S. We repeatedly perform the above procedure until all vertices in the center set, denoted S^*, are v_i for some i. It is clear that S^* is a dominating set of G with $|S^*| \leq |S|$. Thus $OPT' = |V'| - |S| \leq n^{C+1} - |S^*| = n - |S^*| + n(n^C - 1) \leq OPT + n(n^C - 1)$. Combined with the previous result, we have $OPT' = OPT + n(n^C - 1)$, thus completing the reduction and concluding the NP-hardness of $(\geq n^{1-\epsilon})$-SSF.

We now show that the term $n^{1-\epsilon}$ is actually optimal assuming the following Exponential Time Hypothesis (ETH) [12].

Conjecture 1 (Exponential Time Hypothesis [12]). *Any deterministic algorithm for 3-SAT has time complexity $2^{\Omega(n)}$. Consequently, any NP-hard problem has time complexity $2^{n^{\Omega(1)}}$.*

Theorem 3. *Assuming ETH, $(\geq \delta(n))$-SSF is not NP-hard if $\delta(n) = \omega(n^{1-\epsilon})$ for every constant $\epsilon > 0$.*

Proof. Assume $\delta(n) = \omega(n^{1-\epsilon})$ for every constant $\epsilon > 0$. Let G be a graph of order n and minimum degree $\delta(n)$. Let $\lambda(n) = n/\delta(n)$. Then $\lambda(n) = 2^{o(\log n)}$. By Lemma 1 we know that G has a dominating set of size $n \cdot O(\frac{\log \delta(n)}{\delta(n)}) = \lambda(n) \cdot O(\log \delta(n)) \leq 2^{o(\log n)} \cdot O(\log n) = n^{o(1)}$. Therefore, we can enumerate all possible subsets of size $n^{o(1)}$ to find the minimum dominating set of G and hence the maximum spanning star forest of G. This takes time $O(n^{n^{o(1)}}) = O(2^{n^{o(1)} \cdot \log n}) = 2^{n^{o(1)}}$. However, assuming ETH every NP-hard problem has time complexity $2^{n^{\Omega(1)}}$, concluding the proof.

We end up this section by summarizing the obtained results in Table 1.

Table 1. Complexity results for $(\geq \delta(n))$-SSF

$\delta(n)$	complexity of $(\geq \delta(n))$-SSF
$1 \leq \delta(n) \leq O(1)$	NP-hard; APX-complete
$\omega(1) \leq \delta(n) \leq O(n^{1-\epsilon})$ for some fixed $\epsilon > 0$	NP-hard; admits a PTAS
$\omega(n^{1-\epsilon}) \leq \delta(n)$ for any fixed $\epsilon > 0$	not NP-hard assuming ETH

3 Spanning k-Tree Forest Problem

In this section we investigate the node-weighted spanning k-tree forest problem, where k is a fixed positive integer. Recall that the objective of this problem is, given a graph G, to find a spanning forest of G, each component of which has depth at most k when one of its vertices is designated as the root, such that the total weight of all non-root vertices is maximized. As before, we assume that the input graph is connected, since otherwise we can run our algorithm on every connected component of it.

3.1 Improved Approximation Algorithm

Liao and Zhang [13] show that a simple algorithm (denoted as Simple-Alg), based on the idea of dividing a spanning tree into k disjoint star forests, can find a spanning k-tree of weight at least $\frac{k}{k+1}$ times the total weight of all vertices in the input graph.

We first observe that the spanning k-tree forest problem is the complement of the k-domination problem, where the objective of the latter is to find a minimum-weight vertex set of a graph G such that every vertex in G is at distance at most k from some node in this set.

Lemma 2. *For every node-weighted graph G, the maximum weight of a spanning k-tree forest of G equals the total weight of vertices in G minus the minimum weight of a k-domination set of G.*

Proof. Let $G = (V, E)$ and $w : V \to \mathbb{R}^+ \cup \{0\}$ be the weight function on vertices of G. If there is a spanning k-tree forest of G of weight W, the roots in this forest form a k-domination set of weight $\sum_{v \in V} w(v) - W$. If there is a k-domination set of G of weight W', we can perform a breadth-first search rooted at the vertices in this set to construct a spanning k-tree forest of G, which has weight $\sum_{v \in V} w(v) - W'$. □

The following lemma is straightforward and we omit its (trivial) proof.

Lemma 3. *For every graph G, let G^k denote the graph with the same vertex set with G such that every two vertices are adjacent in G^k if and only if they are within distance k from each other in G. Then, every k-domination set of G is a dominating set of G^k, and vice versa.*

It is easy to see that G^k has minimum degree at least k for any connected graph G with at least $k+1$ vertices. Thus the spanning k-tree forest problem is reducible to the ($\geq k$)-SSF problem. Note that the Simple-Alg is a $(\frac{k}{k+1})$-approximation for the spanning k-tree forest problem, which is better than the performance guarantee of $1 - \Theta\left(\frac{\log k}{k}\right)$ for ($\geq k$)-SSF obtained before. We next show how to improve this approximation ratio slightly. The following lemma is due to Chen et. al. [7].

Lemma 4 (Lemma 1 in [7], restated). *There is an $(1-r)^{\frac{1}{r}-1}$-approximation algorithm for node-weighted SSF, where r is the ratio of the weight of the optimal solution over the total weight of all vertices in the graph.*

Algorithm 1. Finding approximate spanning k-tree forest of G

1: Construct G^k from G.
2: Run Rounding-Alg on G^k and Simple-Alg on G, and return the better solution.

We call the algorithm ensured by Lemma 4 Rounding-Alg (since it is based on an LP-rounding scheme). Consider now the algorithm described in Algorithm 1 for the spanning k-tree forest problem given the input graph G. Clearly this algorithm runs in polynomial time. Suppose the total weight of all vertices in G is W, and the weight of the optimum solution is OPT. Let $r = \frac{OPT}{W}$. We have $r \geq \frac{k}{k+1}$, since Simple-Alg returns a solution of weight $\frac{kW}{k+1}$. Note that Simple-Alg has an approximation factor of $\frac{kW}{k+1}/OPT = \frac{k}{(k+1)r}$. The algorithm Rounding-Alg produces a solution with approximation ratio $(1-r)^{\frac{1}{r}-1}$. By comparing the two algorithms and pick the better one, we obtain the following theorem.

Theorem 4. *Algorithm 1 is a $\frac{k+1}{k+2}$-approximation algorithm for the node-weighted spanning k-tree forest problem.*

Proof. It suffices to show that

$$\min_{r:\frac{k}{k+1}\le r<1} \max\left\{\frac{k}{(k+1)r}, (1-r)^{\frac{1}{r}-1}\right\} \ge \frac{k+1}{k+2}.$$

When $r \le \frac{k(k+2)}{(k+1)^2}$ it holds that $\frac{k}{(k+1)r} \ge \frac{k+1}{k+2}$, so we only need to prove it for $r > \frac{k(k+2)}{(k+1)^2}$. As $(1-r)^{\frac{1}{r}-1}$ is monotone increasing with r [7], it suffices to prove

$$\begin{aligned}
&\left(1-\frac{k(k+2)}{(k+1)^2}\right)^{\frac{(k+1)^2}{k(k+2)}-1} \ge \frac{k+1}{k+2}\\
\Leftarrow &\left(\frac{1}{(k+1)^2}\right)^{\frac{1}{k^2+2k}} \ge \frac{k+1}{k+2}\\
\Leftarrow\ & k+1 \le \left(\frac{k+2}{k+1}\right)^{\frac{k^2+2k}{2}}\\
\Leftarrow\ & \ln(k+1) \le \frac{k^2+2k}{2}\ln\left(\frac{k+2}{k+1}\right).
\end{aligned}$$

When $k=2$ it can be directly verified by simple calculation. For $k \ge 3$, using the inequality $\ln(1+x) \ge \frac{x}{1+x}$ for all $x \ge 0$, we have

$$\ln(k+1) \le \frac{k}{2} \le \frac{k^2+2k}{2}\cdot\frac{\frac{1}{k+1}}{\frac{k+2}{k+1}} \le \frac{k^2+2k}{2}\ln\left(\frac{k+2}{k+1}\right).$$

This completes the proof of Theorem 4.

3.2 Inapproximability Results

We next show an inapproximability ratio of $1-\Omega(\frac{1}{k})$ for the spanning k-tree forest problem, which asymptotically matches our approximation ratio of $1-\frac{1}{k+2}$. This result holds even for the unweighted case.

Theorem 5. *There exists a universal constant $c>0$ such that, for every integer $k \ge 2$, it is NP-hard to approximate the spanning k-tree forest problem within a factor of $1-\frac{c}{k}$.*

Proof. Let $\epsilon > 0$ be a constant such that approximating SSF within $1-\epsilon$ is NP-hard [14]. Let $k \ge 2$. Given an instance $G=(V,E)$ of SSF, we construct a corresponding instance G' of the spanning k-tree forest problem (k-SF for short) as follows. Initially set $G'=\emptyset$. For every vertex $v \in V$, add k vertices

$v^{(0)}, v^{(1)}, \ldots, v^{(k-1)}$ to G' together with the edges $(v^{(i)}, v^{(i+1)})$ for all $0 \leq i < k-1$ (i.e., create a chain of length $k-1$). For every edge $e = (u, v) \in E$, add to G' an edge $(u^{(0)}, v^{(0)})$. This finishes the construction of G'.

We claim that G has a spanning star forest of size m if and only if G' has a spanning k-tree forest of size $m + n(k-1)$, n being the number of vertices in G. First assume G has a spanning star forest of size m. By attaching to every vertex the corresponding chain of length $k-1$, we can easily construct a spanning k-tree forest in G' of size $m + n(k-1)$. More formally, letting the edge set of the spanning star forest of G be $\{(u_1, v_1), \ldots, (u_m, v_m)\}$, the edges in $\{(u_1^{(0)}, v_1^{(0)}), \ldots, (u_m^{(0)}, v_m^{(0)})\} \cup \bigcup_{v \in V} \{(v^{(0)}, v^{(1)}), \ldots, (v^{(k-2)}, v^{(k-1)})\}$ form a spanning k-tree forest of G' of size $m + n(k-1)$.

Now suppose G' has a spanning k-tree forest F of size m'. For notational simplicity we call i the order of $v^{(i)}$. We first prove that there exists a spanning k-tree forest of size at least m' such that every center has order 0. Suppose $v^{(t)}$ is a center in F of order $t > 0$. If $v^{(0)}$ is also a center in F, the connected component of F containing $v^{(t)}$ must be a chain, and there exists i s.t. $(v^{(i)}, v^{(i+1)}) \notin F$. We merge this component with that containing $v^{(0)}$; that is, we attach the chain $(v^{(1)}, v^{(2)}, \ldots, v^{(k-1)})$ to $v^{(0)}$. It is easy to verify that the resulting subgraph is also a spanning k-tree forest, and has size at least $m' + 1$. If $v^{(0)}$ is not a center in F, we declare it as a new center, delete from F the (unique) edge incident on $v^{(0)}$, and connect the entire chain $(v^{(1)}, v^{(2)}, \ldots, v^{(k-1)})$ to $v^{(0)}$. This generates a spanning k-tree forest of size at least m'. Hence, w.l.o.g. we may assume that every center in F has order 0.

Let $v^{(0)}$ be any order-0 vertex in G'. If $v^{(0)}$ is neither a center nor directly connected to a center in F, there must exist some i such that $(v^{(i)}, v^{(i+1)})$ is not in F; otherwise, since $v^{(0)}$ is at distance at least 2 from any center in F, $v^{(k-1)}$ is at distance at least $k+1$ from any center in F, contradicting the k-tree property. Let $u^{(0)}$ be the vertex to which $v^{(0)}$ is connected in F (such vertex exists since $v^{(0)}$ is not a center and every center is of order 0). We delete $(v^{(0)}, u^{(0)})$ from F and designate $v^{(0)}$ as a new center in F. We then add all edges $(v^{(i)}, v^{(i+1)})$ to F (at least 1 new edge is added). Recursively doing this will finally give a spanning k-tree forest F' of size $\geq m'$, in which every vertex of order 0 is either a center itself or is directly connected to another center. Now, deleting edges $(v^{(i)}, v^{(i+1)})$ for all v and i (if they are in F') and then replacing $v^{(0)}$ with v for all $v \in V$ will produce a spanning star forest of G, which has size at least $m' - n(k-1)$. We have thus proved that G has a spanning star forest of size m if and only if G' has a spanning k-tree forest of size $m + n(k-1)$, and one solution can be easily converted to another in polynomial time.

Suppose the k-SF problem admits an r-approximation algorithm. Denoting by m the size of the optimal solution to G, we know that the optimal solution to G' has size $m + n(k-1)$. Also note that $m \geq n/2$ since G is connected. Applying the r-approximation scheme on G' and then transforming it back to a solution to G will give us a spanning star forest of size at least $r(m + n(k-1)) - n(k-1)$. As approximating SSF within $1 - \epsilon$ is NP-hard, there exist an instance G with $n, m > 0$ such that $r(m + n(k-1)) - n(k-1) \leq (1-\epsilon)m$. We thus have

$r \leq \frac{(1-\epsilon)m+n(k-1)}{m+n(k-1)} = \frac{(1-\epsilon)+\frac{n}{m}(k-1)}{1+\frac{n}{m}(k-1)} \leq \frac{1-\epsilon+2(k-1)}{1+2(k-1)} = 1 - \frac{\epsilon}{2k-1} \leq 1 - \frac{\epsilon/2}{k}$. Taking $c = \epsilon/2$ completes the proof. □

We note that Theorem 1 follows directly from Theorem 5, since as argued before, the spanning k-tree forest problem is a special case of the $(\geq k)$-spanning star forest problem.

4 Improved Hardness Results for Spanning Star Forest

In this section we show that, by a careful examine of the reductions used by Chlebík and Chlebíková [8] for proving hardness for NP-hard problems on low-degree instances, we can improve the hardness bound for the unweighted spanning star forest problem.

Theorem 6. *For any $\epsilon > 0$, it is NP-hard to approximate the spanning star forest problem to a factor of $\frac{244}{245} + \epsilon$ even on graphs of maximum degree 5.*

Proof. One of Chlebík and Chlebíková's results says that vertex cover in 4-regular graphs cannot be approximated to within $\frac{52}{53}$ assuming $P \neq NP$ (Corollary 18 in [8]). The reductions in their paper crucially depend on a class of carefully constructed gadgets called *consistency gadgets*. Revisiting their proof of Theorem 17 in [8], it can be seen that they actually prove the following stronger result: Given any 4-regular graph G of n vertices, it is NP-hard to tell whether the optimal vertex cover has size at most $n\frac{2(V_H-M_H)/k+8+\epsilon}{2V_H/k+12}$, or has size at least $n\frac{2(V_H-M_H)/k+9-\epsilon}{2V_H/k+12}$, where $\epsilon > 0$ is any small constant, H is a fixed consistency gadget (depending on ϵ) and V_H, M_H, k are parameters of H satisfying that $V_H = 2M_H$ and $M_H/k \leq 21.7$.

Given a graph $G = (V, E)$, we construct a graph G' by adding a vertex v_e to V for every $e \in E$, and then connecting v_e and the two endpoints of e. More specifically, $G' = (V', E')$ where $V' = V \cup \{v_e | e \in E\}$ and $E' = E \cup \{(x, v_e), (y, v_e) | e = (x, y) \in E\}$. It is easy to argue that the size of the optimum vertex cover of G is equal to that of the optimum dominating set of G', which equals $|V'|$ minus the size of the maximum spanning star forest of G'. Combining this reduction with the aforementioned gap instances of 4-regular vertex cover, it is easy to argue the following: Given any graph G of $3n$ vertices with maximum degree 5, it is NP-hard to distinguish whether $OPT_{ssf}(G) \leq n\left(3 - \frac{2(V_H-M_H)/k+9-\epsilon}{2V_H/k+12}\right)$ or $OPT_{ssf}(G) \geq n\left(3 - \frac{2(V_H-M_H)/k+8+\epsilon}{2V_H/k+12}\right)$, where $OPT_{ssf}(G)$ denote the maximum size of a spanning star forest of G. Hence, it is NP-hard to approximate unweighted SSF to a factor of

$$\left(3 - \frac{2(V_H - M_H)/k + 9 - \epsilon}{2V_H/k + 12}\right) / \left(3 - \frac{2(V_H - M_H)/k + 8 + \epsilon}{2V_H/k + 12}\right)$$
$$= \frac{27 + 4V_H/k + 2M_H/k + \epsilon}{28 + 4V_H/k + 2M_H/k - \epsilon} \leq \frac{27 + 10 \cdot 21.7 + \epsilon}{28 + 10 \cdot 21.7 - \epsilon} \leq \frac{244}{245} + \epsilon,$$

for any small $\epsilon > 0$, since $V_H = 2M_H$ and $M_H/k \leq 21.7$. The theorem is thus proved.

References

1. Agra, A., Cardoso, D., Cerfeira, O., Rocha, E.: A spanning star forest model for the diversity problem in automobile industry. In: Proceedings of the 17th European Conference on Combinatorial Optimization, ECCO XVII (2005)
2. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley, New York (1992)
3. Arora, S., Karger, D., Karpinski, M.: Polynomial time approximation schemes for dense instances of NP-hard problems. J. Comput. Syst. Sci. 58(1), 193–210 (1999)
4. Athanassopoulos, S., Caragiannis, I., Kaklamanis, C., Kyropoulou, M.: An improved approximation bound for spanning star forest and color saving. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 90–101. Springer, Heidelberg (2009)
5. Berry, V., Guillemot, S., Nicolas, F., Paul, C.: On the approximation of computing evolutionary trees. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 115–125. Springer, Heidelberg (2005)
6. Chakrabarty, D., Goel, G.: On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. In: Proceedings of 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 687–696 (2008)
7. Chen, N., Engelberg, R., Nguyen, C.T., Raghavendra, P., Rudra, A., Singh, G.: Improved approximation algorithms for the spanning star forest problem. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 44–58. Springer, Heidelberg (2007)
8. Chlebík, M., Chlebíková, J.: Complexity of approximating bounded variants of optimizatin problems. Theor. Comput. Sci. 354, 320–338 (2006)
9. Duh, R., Furer, M.: Approximation of k-set cover by semi local optimization. In: Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC), pp. 256–264 (1997)
10. Feige, U.: A threshold of $\ln n$ for aproximating set cover. J. ACM 45(4), 634–652 (1998)
11. He, J., Liang, H.: An improved approximation algorithm for spanning star forest in dense graphs. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part II. LNCS, vol. 6509, pp. 160–169. Springer, Heidelberg (2010)
12. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. 62(2), 367–375 (2001)
13. Liao, C.-S., Zhang, L.: Approximating the spanning k-tree forest problem. In: Deng, X., Hopcroft, J.E., Xue, J. (eds.) FAW 2009. LNCS, vol. 5598, pp. 293–301. Springer, Heidelberg (2009)
14. Nguyen, C.T., Shen, J., Hou, M., Sheng, L., Miller, W., Zhang, L.: Approximating the spanning star forest problem and its applications to genomic sequence alignment. SIAM J. Comput. 38(3), 946–962 (2008)
15. Vazirani, V.: Approximation Algorithms. Springer, Heidelberg (2001)