Noname manuscript No.
(will be inserted by the editor)

Improved Approximation for Spanning Star Forest in Dense Graphs

Jing He · Hongyu Liang

the date of receipt and acceptance should be inserted later

Abstract A spanning subgraph of a graph G is called a *spanning star forest* of G if it is a collection of node-disjoint trees of depth at most 1. The *size* of a spanning star forest is the number of leaves in all its components. The goal of the *spanning star forest problem* is to find the maximum-size spanning star forest of a given graph.

In this paper, we study the spanning star forest problem on c-dense graphs, where for any fixed $c \in (0,1)$, a graph of n vertices is called *c-dense* if it contains at least $cn^2/2$ edges. We design a $(\alpha + (1-\alpha)\sqrt{c} - \epsilon)$-approximation algorithm for spanning star forest in c-dense graphs for any $\epsilon > 0$, where $\alpha = \frac{193}{240}$ is the best known approximation ratio of the spanning star forest problem in general graphs. Thus, our approximation ratio outperforms the best known bound for this problem when dealing with c-dense graphs. We also prove that, for any constant $c \in (0,1)$, approximating spanning star forest in c-dense graphs is APX-hard. We then demonstrate that for weighted versions (both node- and edge- weighted) of this problem, we cannot get *any* approximation algorithm with strictly better performance guarantee on c-dense graphs than on general graphs. Finally, we give strong inapproximability results for a closely

This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, and the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174. Part of this work was done while the authors were visiting Cornell University.

Jing He
Institute for Interdisciplinary Information Sciences, FIT 4-609, Tsinghua University, Beijing, China
E-mail: he-j08@mails.tsinghua.edu.cn

Hongyu Liang
Corresponding author
Institute for Interdisciplinary Information Sciences, FIT 4-609, Tsinghua University, Beijing, China
Tel.: +86-13552139521
E-mail: lianghy08@mails.tsinghua.edu.cn

related problem, namely the minimum dominating set problem, restricted on c-dense graphs.

Keywords spanning star forest · dense graph · approximation algorithm · hardness of approximation

1 Introduction

We consider the spanning star forest problem. A graph is called a *star* if it can be regarded as a tree of depth at most 1, or equivalently, there is one vertex (called the *center*) adjacent to all other vertices (called *leaves*) in the graph. A single node is by definition also a star. A *star forest* is a forest whose connected components are all stars. The *size* of a star forest is the number of its leaves. 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), introduced in [13], is the problem of finding a spanning star forest of maximum size in a given graph. This problem has found applications in various areas. Nguyen et al. [13] use it as a subroutine to design an algorithm for aligning multiple genomic sequences, which is an important bioinformatics problem in comparative genomics. This model has also been applied to the comparison of phylogenetic trees [4] and the diversity problem in the automobile industry [1].

It is easy to see that there is a one-one correspondence between spanning star forests and dominating sets of a given graph. A dominating set of a graph G is a subset of vertices D such that every vertex not in D is adjacent to at least one vertex in D. The *minimum dominating set problem* is to find a smallest dominating set of a given graph. Given a spanning star forest of G, it is easy to argue that the collection of its all centers is a dominating set of G, and the size of the spanning star forest is equal to the number of vertices in G minus the size of the corresponding dominating set. On the other hand, given a dominating set of G, we can construct a spanning star forest of G whose centers are exactly those vertices in the dominating set. Thus, the two problems are equivalent in finding the optimum solution. We also call one problem the *complement* of another, following the notion used before [3, 8].

However, the two problems appear totally different when the approximability is considered. By Feige's famous result [10], the dominating set problem cannot be approximated within factor $(1-\epsilon)\ln n$ in polynomial time for any fixed $\epsilon > 0$ unless $NP \subseteq DTIME(n^{O(\log\log n)})$. In contrast, a fairly simple algorithm with the idea of dividing a spanning tree into alternating levels gives a 0.5-approximation to the spanning star forest problem. Nguyen et al. [13] proposed a 0.6-approximation algorithm using the fact that every graph of n vertices of minimum degree 2 has a dominating set of size at most $\frac{2}{5}n$ except for very few special cases which can be enumerated. In addition, they proved that it is NP-hard to approximate the problem to any factor larger than $\frac{259}{260}$. They also introduced the edge-weighted version of this problem, whose objective is to find a spanning star forest in which the total weight of edges is

maximized, and showed a 0.5-approximation algorithm for this variant. Later on, the approximation ratio for unweighted SSF was improved to 0.71 by Chen et al. [8] based on solving a natural linear programming relaxation combined with a randomized rounding stage. They also considered another generalization of SSF where each node has a non-negative weight and the objective is to find a spanning star forest in which the total weight of all leaves is maximized. Note that node-weighted SSF is just the complement of the weighted dominating set problem where each vertex has a weight and the goal is to find a minimum-weight dominating set of the given graph. For this version, they showed that a similar algorithm achieves an approximation factor of 0.64. Athanassopoulos et al. [3] realized that the unweighted spanning star forest problem is actually a special case of the complementary set cover problem, and designed a 0.804-approximation for it (also for complementary set cover) using the idea of semi-local search for k-set cover [9]. Regarding the hardness results, it was proved by Chakrabarty and Goelin [7] that edge-weighted SSF and node-weighted SSF cannot be approximated to $\frac{10}{11} + \epsilon$ and $\frac{13}{14} + \epsilon$ respectively, unless $P = NP$.

1.1 Our contributions

We study variants of the spanning star forest problem in c-dense graphs. A graph of n vertices is called *c-dense*, for some constant $c \in (0, 1)$, if it contains at least $cn^2/2$ edges [2]. One can show by a simple probabilistic argument that almost all graphs are dense. Thus, it captures many real-world models. In fact, this setting has received extensive studies for various combinatorial problems like vertex cover, max-cut, Steiner tree, minimum maximal matching, etc. (see [2,6,11,12,15]). To our knowledge, ours is the first study on the spanning star forest problem in the class of c-dense graphs.

We first design an approximation algorithm for (unweighted) spanning star forest in c-dense graphs with an approximation ratio better than the previously best known ratio of this problem in general graphs, for any $c \in (0, 1)$. More precisely, denoting by $\alpha = \frac{193}{240} (\approx 0.804)$ the best known approximation ratio for spanning star forest [3], our algorithm achieves an approximation factor of $\alpha + (1 - \alpha)\sqrt{c} - \epsilon$, for any $\epsilon > 0$. Note that this factor is larger than 0.9 whenever $c \geq 0.25$, and is larger than 0.96 when $c \geq 0.64$. Thus, it is a quite strong performance guarantee. Our algorithm consists of two stages. The first stage is actually a greedy procedure that chooses the vertex covering the largest number of uncovered vertices, and adds it to a maintained dominating set of the input graph. It stops when the number of uncovered vertices is smaller than some prespecified threshold, and goes to the second stage. In this stage, we find a set of vertices dominating the uncovered ones by reducing it to a problem called *complementary partial dominating set*, which will be formally defined in Section 2.2. We will show in Section 2.2 that this problem can be approximated as well as the complementary set cover problem considered in [3]. Combining the two stages, we find a dominating set of the graph of

relatively small size, and then construct a spanning star forest in the standard way, which can be proved to be a good approximation to the problem.

We then prove that the spanning star forest problem in c-dense graphs is APX-hard. Specifically, we prove that for any $c \in (0,1)$, there exists $\epsilon = \epsilon(c) > 0$ such that approximating SSF in c-dense graphs to within a factor of $1-\epsilon$ is NP-hard. Thus, the technique developed by Arora et al. [2] for designing PTAS for combinatorial problems in dense instances cannot be applied to our problem.

Next we consider the weighted versions (both node- and edge-weighted) of this problem. A little surprisingly, we show that *any* approximation algorithm for weighted spanning star forest in c-dense graphs does not guarantee an approximation ratio strictly larger than that in general graphs. This is proved by an (almost) approximation-preserving reduction from general instances of this problem to c-dense instances.

Finally, we show that the dominating set problem in c-dense graphs shares the same inapproximability result with that in general graphs. Thus, the $(1+\ln n)$-approximation achieved by a greedy approach is nearly the best we can hope for. This again shows that the spanning star forest problem and the dominating set problem are very different regarding the approximability, although they are equivalent in exact optimization.

1.2 Notation used for approximation algorithms

For $\beta \in (0,1)$ (resp. $\beta > 1$) and a maximization (resp. minimization) problem Π, an algorithm is called a *β-approximation algorithm* for Π if given an instance $\mathcal{I}$ of Π, it runs in polynomial time and produces a solution with objective value at least (resp. at most) $\beta \cdot OPT(\Pi, \mathcal{I})$, where $OPT(\Pi, \mathcal{I})$ denotes the objective value of the optimum solution to the instance $\mathcal{I}$ of the problem Π. The value β is also called the *approximation ratio*, *approximation factor*, or *performance guarantee* of the algorithm for the problem Π. Moreover, β can be a function of the input size or some parameters in the input. We say the problem 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. For standard definitions and notations not given here, we refer the readers to [16].

2 Complementary Partial Dominating Set

In this section, we introduce the *complementary partial dominating set problem*, which is useful for designing our algorithm for spanning star forest in dense graphs. Before presenting its formal definition, we need to mention another related problem called the *complementary set cover problem*.

2.1 Complementary set cover

We briefly review the complementary set cover problem (CSC for short) [3], since some results of it will be used later. The input of CSC is a pair $(\mathcal{S}, U)$, which consists of a ground set U of elements and a set $\mathcal{S}$ containing some subsets of U. The set $\mathcal{S}$ is guaranteed to be close under subsets, that is, for any $S \in \mathcal{S}$ and $S' \subseteq S$, we have $S' \in \mathcal{S}$. The representation of $\mathcal{S}$ can be implicit, e.g., only inclusion-wise maximal sets in it are specified. The goal is to find a collection of pairwise-disjoint subsets $S_1, S_2, \ldots, S_k \in \mathcal{S}$ whose union is U, such that $|U| - k$ is maximized. It is shown in [3] that CSC has a $\frac{193}{240}$-approximation algorithm, which only selects subsets of size at most 6.

2.2 Complementary partial dominating set

Let $G = (V, E)$ be a simple undirected graph. For any vertex $v \in V$, let $N[v] = \{u \in V : (u, v) \in E\} \cup \{v\}$ be the neighborhood of v when regarding v as a neighbor of itself. Let $N[U] = \bigcup_{v \in U} N[v]$ for $U \subseteq V$. For two subsets $U_1, U_2 \subseteq V$, we say U_1 *dominates* U_2, or U_1 is a *dominating set* of U_2, if $U_2 \subseteq N[U_1]$. The *complementary partial dominating set problem* (CPDS for short) is defined as follows.

Input: A graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$.

Output: A set $U \subseteq V$ that dominates V' such that $|V'| - |U|$ is maximized.

Although the objective we use seems to be equivalent to finding the minimum-size dominating set of V', they are totally different when considering the approximability. It is easy to see that the minimization version of CPDS generalizes the dominating set problem and thus cannot be approximated to within $\gamma \log n$ for some constant γ unless $P = NP$ [10,14], while as is shown below, CPDS allows a constant factor approximation algorithm.

Theorem 1 *There is a $\frac{193}{240}$-approximation algorithm for CPDS.*

Proof Given an instance $\mathcal{I} = (G, V')$ of CPDS, we regard it as an instance $\mathcal{I}' = (\mathcal{S}, U)$ of CSC in the following way. The ground set U is just V', and $\mathcal{S}$ contains all subsets of V' each of which is dominated by some vertex in V, i.e. $\mathcal{S} = \{W \subseteq V' : \exists v \in V \text{ s.t. } W \subseteq N[v]\}$. It is easy to see that $\mathcal{S}$ is close under subsets. (Note that $\mathcal{S}$ may have exponential size; we will come back to this point later.) Now, given a solution to the instance $\mathcal{I}$ of CPDS with objective value s, we can easily construct a solution to the instance $\mathcal{I}'$ of CSC with no smaller objective value, and vice versa. Therefore, the two instances have a same optimal objective value, and we can apply the $\frac{193}{240}$-approximation algorithm for CSC on $\mathcal{I}'$ to obtain a solution to $\mathcal{I}$ with the same approximation ratio. However, the instance $\mathcal{I}'$ may have exponential size since it may contain all subsets of V'. To overcome this, we just note that the $\frac{193}{240}$-approximation algorithm for CSC only deals with sets in $\mathcal{S}$ of size at most 6, and all subsets of V' of size at most 6 can surely be enumerated in polynomial time. □

3 Algorithm Description and Analysis

In this section, we give an approximation algorithm for the spanning star forest problem in dense graphs. Fix $c \in (0,1)$. Let $\alpha = \frac{193}{240}$ be the best known approximation ratio for CPDS. Let ϵ be any constant such that $0 < \epsilon < \sqrt{c}$. Let $\delta = 1 - \sqrt{c} + \epsilon$, $M = 2/(c - (\sqrt{c} - \epsilon)^2)$, and $N_0 = M/(\epsilon(1-\delta))$. Note that δ, M and N_0 are all positive constants only depending on c and ϵ.

Algorithm 1 Approximate SSF in c-dense graphs

Input: A c-dense graph $G = (V, E)$.
Output: A spanning star forest of G.

If $n \leq N_0$ we perform the exhaustive search to get the optimal solution. In the following we assume $n > N_0$.
$A \leftarrow \emptyset$, $B \leftarrow \emptyset$, $C \leftarrow V$.
Stage 1:
while $|C| \geq \delta n$ **do**
  Find the vertex $v \in B \cup C$ that dominates the largest number of vertices in C.
  Set $A \leftarrow A \cup \{v\}$, $B \leftarrow N[A] \setminus A$, and $C \leftarrow V \setminus N[A]$.
end while
Stage 2:
Construct an instance $\mathcal{I} = (G', V')$ of CPDS, where G' is the subgraph of G induced on the vertex set $B \cup C$, and $V' = C$. Run the α-approximation algorithm for CPDS on $\mathcal{I}$ to get a dominating set of C, denoted by S.
return a spanning star forest rooted on $A \cup S$.

We present our algorithm for SSF in c-dense graphs as Algorithm 1. Note that at the beginning (and the end) of every execution of the WHILE loop, A, B and C form a partition of V. To show that the obtained star forest is large, we bound the cardinality of A and S respectively.

Lemma 1 *At the end of Stage 1, it holds that* $|A| \leq M$.

Proof Consider the moment right before some vertex v is added to A. Due to the loop condition, we have $|C| \geq \delta n$, and $|A \cup B| = n - |C| \leq (1-\delta)n$. Thus, the number of edges in E with both endpoints in $A \cup B$ is at most $((1-\delta)n)^2/2$. Since $|E| \geq cn^2/2$, the number of edges in E with at least one endpoint in C is at least $cn^2/2 - ((1-\delta)n)^2/2 = n^2/M$. Let E_1 be the set of edges with one endpoint in B and another in C, and E_2 be the set of edges with both endpoints in C. Note that the previous statement is equivalent to $|E_1| + |E_2| \geq n^2/M$, since by definition there are no edges between A and C.

For any vertex $v \in B \cup C$, let $D(v) = N[v] \cap C$ be the set of vertices in C dominated by v. Consider $D = \sum_{v \in B \cup C} |D(v)|$. It is easy to see that every edge in E_1 contributes 1 to this sum, while each edge in E_2 contributes 2. Hence, $D = |E_1| + 2|E_2| \geq n^2/M$, from which we know that there exists a vertex $v^* \in B \cup C$ such that $|D(v^*)| \geq n/M$. Note that the greedy step in the algorithm is just to pick the vertex v with the largest $|D(v)|$. Therefore, after adding v to A and updating B and C correspondingly, the size of $A \cup B$

increases by at least n/M. Since there are only n vertices, we can add at most M of them to A, completing the proof of Lemma 1. □

Lemma 2 *$|S| \leq \delta(1-\alpha)n + \alpha k$, where k is the size of the smallest subset $U \subseteq B \cup C$ that dominates C.*

Proof By the definition of CPDS, we know that the value of the optimum solution to its instance $\mathcal{I}$ defined in Algorithm 1 is precisely $|C| - k$. As the solution S is obtained by applying the α-approximation algorithm for CPDS, we have $|C|-|S| \geq \alpha(|C|-k)$. Rearranging terms gives $|S| \leq (1-\alpha)|C|+\alpha k \leq \delta(1-\alpha)n+\alpha k$, where the second inequality follows from the fact that $|C| \leq \delta n$ at the end of Stage 1. □

We are ready to prove our main theorem.

Theorem 2 *Algorithm 1 is a $(\alpha + (1-\alpha)\sqrt{c} - 2\epsilon)$-approximation algorithm for the spanning star forest problem in c-dense graphs.*

Proof Clearly Algorithm 1 runs in polynomial time. Furthermore, it finds the optimal spanning star forest of G when $n \leq N_0$, and produces a solution of size $n - |A| - |S| \geq (1 - \delta(1-\alpha))n - \alpha k - M$ when $n > N_0$, by Lemmas 1 and 2. The size of the optimal solution is $n - k^*$, where k^* is the size of the smallest dominating set of G. It is easy to see that k^* is not smaller than the size of the smallest subset of V that dominates C. Since no edges exist between A and C, the latter quantity is equal to k, the size of the smallest subset of $B \cup C$ that dominates C. Therefore, we have $n - k^* \leq n - k$. We also note that $k \leq |C| \leq \delta n$ since C dominates itself. The approximation ratio of Algorithm 1 can thus be bounded from below by

$$\begin{aligned}
&\frac{(1-\delta(1-\alpha))n - \alpha k - M}{n - k^*} \\
&\geq \frac{(1-\delta(1-\alpha))n - \alpha k - M}{n - k} \\
&= \alpha + \frac{(1-\alpha)(1-\delta)n}{n-k} - \frac{M}{n-k} \\
&\geq \alpha + (1-\alpha)(1-\delta) - \frac{M}{n-\delta n} \\
&\geq \alpha + (1-\alpha)(\sqrt{c}-\epsilon) - \frac{M}{(1-\delta)N_0} \\
&\geq \alpha + (1-\alpha)\sqrt{c} - 2\epsilon,
\end{aligned}$$

which concludes the proof of Theorem 2. □

4 Hardness Results

We now show that for every $0 < c < 1$, SSF in c-dense graphs does not admit a polynomial-time approximation scheme, unless $P = NP$. Thus, the technique

developed by Arora et al. [2] for designing PTAS for combinatorial problems in dense instances cannot be applied to this problem.

Theorem 3 *For any constant $c \in (0,1)$, there exists a constant $\epsilon = \epsilon(c) > 0$, such that it is NP-hard to approximate the spanning star forest problem in c-dense graphs to a factor of $1-\epsilon$.*

Proof We reduce the general SSF problem to SSF in c-dense graphs. Let $G = (V, E)$ be an input to general SSF. Let $n = |V|$, $k = \lceil 2\sqrt{c}/(1-\sqrt{c}) \rceil$, and let OPT denote the size of the largest spanning star forest of G. It is easy to verify that $k > \sqrt{c}(k+1)$. We assume w.l.o.g. that $n \geq k/(k^2 - c(k+1)^2) > 0$, since otherwise we can just do a brute-force search for the constant-size (note that k and c are both constants) input graph. We also assume that G is connected, since connected and disconnected versions of general SSF share the same hardness-of-approximation result. We thus have $OPT \geq n/2$, since any connected graph on n vertices has a dominating set of size at most $n/2$. Let H be a complete graph on a vertex set of size kn which is disjoint from V, and let $G' = G \cup H$.

We verify that G' is c-dense. As G' has $n' = (k+1)n$ vertices and at least $kn(kn-1)/2$ edges, it suffices to show that $kn(kn-1)/2 \geq c(k+1)^2 n^2/2$, or $n \geq k/(k^2 - c(k+1)^2)$, which is exactly our assumption on n. Since G' consists of two disjoint components, it is clear that $OPT' = OPT + kn - 1$, OPT' denoting the size of the largest spanning star forest of G'. Moreover, given a spanning star forest of G' of size s', we can easily construct a spanning star forest of G of size at least $s' - (kn-1)$. Thus, given any β-approximation algorithm for SSF in c-dense graphs, we can obtain a spanning star forest of G of size $\beta(OPT + kn - 1) - (kn-1)$. On the other hand, we know that there is a constant $\gamma > 0$ such that approximating general SSF within $1-\gamma$ is NP-hard [13]. Therefore, there exists G such that $\beta(OPT + kn - 1) - (kn-1) \leq (1-\gamma)OPT$, from which we derive that

$$\begin{aligned}
\beta &\leq \frac{(1-\gamma)OPT + kn - 1}{OPT + kn - 1} \\
&= 1 - \gamma + \frac{\gamma(kn-1)}{OPT + kn - 1} \\
&\leq 1 - \gamma + \frac{\gamma(kn-1)}{n/2 + kn - 1} \\
&< 1 - \gamma + \frac{\gamma k}{k + 1/2}.
\end{aligned}$$

The proof is completed by choosing $\epsilon = \gamma/(2k+1)$. □

We have designed an algorithm for SSF in c-dense graphs whose approximation ratio outperforms the best known bound for general SSF, for every $0 < c < 1$. A natural question is whether we can generalize our technique to weighted versions of SSF. A little surprisingly, we show in the following that this is not the case: We cannot design *any* approximation algorithm for node-

(resp. edge-)weighted SSF in c-dense graphs with a strictly larger performance guarantee than that of general node- (resp. edge-)weighted SSF.

Theorem 4 *For any constants $c \in (0,1)$ and $\beta, \epsilon > 0$, the existence of a β-approximation algorithm for node- (resp. edge-)weighted SSF in c-dense graphs implies that of a $(\beta - \epsilon)$- (resp. β-)approximation algorithm for node- (resp. edge-)weighted SSF in general graphs.*

Proof The edge-weighted case is easy since we can regard every edge-weighted graph as a complete graph (which is c-dense for any $c < 1$ and large enough n) with some edges having weight 0. Thus, in the following we consider the node-weighted version of SSF. Fix c, ϵ and β. Let $G = (V, E)$ be an input graph to node-weighted SSF, and $w : V \to \mathbb{Q}^+ \cup \{0\}$ be the weight function on its nodes. Let $n = |V|$ and OPT denote the maximum weight of a spanning star forest of G. We assume that $OPT > 0$ since the case of $OPT = 0$ is easily detectable. Let $w^* = \min\{w(v) : v \in V \text{ and } w(v) > 0\}$. Clearly $OPT \geq w^*$. We apply a reduction similar to that used in the proof of Theorem 3 to get a c-dense graph $G' = G \cup H$, with the only difference that we set the weights of all vertices in H to 1, and multiply the weights of all vertices in G by a factor of $\Delta = (1-\beta)(kn-1)/(\epsilon w^*)$ (recall that k is the constant defined in the proof of Theorem 3). Now we have $OPT' = \Delta \cdot OPT + kn - 1$ where OPT' denotes the maximum weight of a spanning star forest of G', and a spanning star forest of G' of weight s' can be easily transformed to a spanning star forest of G of weight at least $(s' - (kn-1))/\Delta$. Thus, given a β-approximation to node-weighted SSF in c-dense graphs, we can design an approximation algorithm for node-weighted SSF in general graphs with an approximation ratio of

$$\begin{aligned}\beta' &\geq \frac{(\beta(\Delta \cdot OPT + kn - 1) - (kn-1))/\Delta}{OPT} \\ &= \beta - \frac{(1-\beta)(kn-1)}{\Delta \cdot OPT} \\ &\geq \beta - \frac{(1-\beta)(kn-1)}{\Delta \cdot w^*} = \beta - \epsilon,\end{aligned}$$

concluding the proof of Theorem 4. □

Finally, we show that the dominating set problem, as the complement of SSF, remains hard to approximate even in dense graphs.

Theorem 5 *For any constants $c \in (0,1)$ and $\epsilon > 0$, there is no $(1-\epsilon)\ln n$-approximation algorithm for dominating set in c-dense graphs, where n is the number of vertices in the input graph, unless $NP \subseteq DTIME(n^{O(\log\log n)})$.*

Proof We show how to use a $(1-\epsilon)\ln n$-approximation for dominating set in c-dense graphs to design a $(1-\epsilon')\ln n$-approximation for dominating set in general graphs, thus proving the theorem since by [10] this implies $NP \subseteq DTIME(n^{O(\log\log n)})$. Given a graph $G = (V, E)$, we first exhaustively check if the optimal dominating set has size at most $\lceil 1/\epsilon \rceil$. If so, we can find it

in polynomial time. Otherwise, we apply a reduction similar to that used in the proof of Theorem 3 to obtain a c-dense graph G'. Denoting by OPT and OPT' the size of the minimum dominating set of G and G' respectively, it is clear that $OPT' = OPT + 1$, and a dominating set of G' of size s can be easily converted to one of G of size at most $s - 1$. Therefore, given a $(1 - \epsilon) \ln n$-approximation for dominating set on c-dense graphs, we can obtain an approximation algorithm for it on general graphs with approximation ratio at most $((1-\epsilon)\ln n(OPT+1)-1)/OPT < (1-\epsilon)\ln n(1+1/OPT) \leq (1-\epsilon^2)\ln n$, since $OPT \geq \lceil 1/\epsilon \rceil$. This finishes the proof of Theorem 5.

5 Conclusion

In this paper, we explored the spanning star forest problem in c-dense graphs, and devised an algorithm with approximation ratio better than the previously best known ratio for this problem in general graphs. We also showed that this problem does not admit a PTAS unless $P = NP$, thus ruling out the possibility of applying the general technique developed by Arora et al. to this problem. We then showed hardness results for its weighted versions as well as its complementary problem, the dominating set problem in dense graphs.

An interesting open question is whether we can generalize the notion of c-dense graphs to allow $c = o(1)$ and still get better approximation than in general graphs. Such graphs have recently been considered by Cardinal et al. [5], on which they obtain tight approximation bounds for several combinatorial problems, including vertex cover, connected vertex cover and the Steiner tree problem. It is also of interests to bridge the gap between algorithmic and hardness results for spanning star forest in c-dense graphs, since the inapproximability factor derived by our reduction is very close to 1.

References

1. A. Agra, D. Cardoso, O. Cerfeira, and E. Rocha. 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. S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *J. Comput. Syst. Sci.*, 58(1):193–210, 1999.
3. S. Athanassopoulos, I. Caragiannis, C. Kaklamanis, and M. Kuropoulou. An improved approximation bound for spanning star forest and color saving. In *34th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 5734 of *LNCS*, pages 90–101, 2009.
4. V. Berry, S. Guillemot, F. Nicholas, and C. Paul. On the approximation of computing evolutionary trees. In *the 11th International Computing and Combinatorics Conference (COCOON)*, volume 3595 of *LNCS*, pages 115–125, 2005.
5. J. Cardinal, M. Karpinski, R. Schmied, and C. Viehmann. Approximating subdense instances of covering problems. arXiv:1011.0078v2, 2010.
6. J. Cardinal, S. Langerman, and E. Levy. Improved approximation bounds for edge dominating set in dense graphs. *Theor. Comput. Sci.*, 410:949–957, 2009.
7. D. Chakrabarty and G. Goel. 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), pages 687–696, 2008.
8. N. Chen, R. Engelberg, C. T. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved approximation algorithms for the spanning star forest problem. In *10th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 4627 of *LNCS*, pages 44–58, 2007.
9. R. Duh and M. Furer. Approximation of k-set cover by semi local optimization. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 256–264, 1997.
10. U. Feige. A threshold of $\ln n$ for aproximating set cover. *J. ACM*, 45(4):634–652, 1998.
11. S. Gaspers, D. Kratsch, M. Liedloff, and I. Todinca. Exponential time algorithms for the minimum dominating set problem on some graph classes. *ACM Trans. Algorithms*, 6(1), 2009.
12. T. Imamura and K. Iwama. Approximating vertex cover on dense graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 582–589, 2005.
13. C. T. Nguyen, J. Shen, M. Hou, L. Sheng, W. Miller, and L. Zhang. Approximating the spanning star forest problem and its applications to genomic sequence alignment. *SIAM J. Comput.*, 38(3):946–962, 2008.
14. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 475–484, 1997.
15. I. Schiermeyer. Problems remaining NP-complete for sparse or dense graphs. *Discuss. Math. Graph Theory*, 15:33–41, 1995.
16. V. Vazirani. *Approximation Algorithms*. Springer, 2001.