# Improved Approximation Algorithms for the Spanning Star Forest Problem

Ning Chen[1⋆], Roee Engelberg[2⋆⋆], C. Thach Nguyen[1],
Prasad Raghavendra[1⋆⋆⋆], Atri Rudra[1†], and Gyanit Singh[1]

[1] Department of Computer Science & Engineering,
University of Washington, Seattle, USA
{ning,ncthach,prasad,atri,gyanit}@cs.washington.edu

[2] Department of Computer Science,
Technion, Haifa, Israel.
roee@cs.technion.ac.il

**Abstract.** A *star* graph is a tree of diameter at most two. A *star forest* is a graph that consists of node-disjoint star graphs. In the spanning star forest problem, given an unweighted graph $G$, the objective is to find a star forest that contains all the vertices of $G$ and has the maximum number of edges. This problem is the complement of the dominating set problem in the following sense: On a graph with $n$ vertices, the size of the maximum spanning star forest is equal to $n$ minus the size of the minimum dominating set.

We present a 0.71-approximation algorithm for this problem, improving upon the approximation factor of 0.6 of Nguyen et al. [9]. We also present a 0.64-approximation algorithm for the problem on node-weighted graphs. Finally, we present improved hardness of approximation results for the weighted versions of the problem.

## 1 Introduction

A *star* graph is a tree of diameter at most two. Equivalently, a star graph consists of a vertex designated *center* along with a set of *leaves* adjacent to it. In particular, a singleton vertex is a star as well. Given an undirected graph, a spanning star forest consists of a set of node-disjoint stars that cover all the nodes in the graph. In the spanning star forest problem, the objective is to maximize the number of edges (or equivalently, leaves) present in the forest.

A dominating set of a graph is a subset of the vertices such that every other vertex is adjacent to a vertex in the dominating set. Observe that in a spanning star forest solution, each vertex is either a center or adjacent to a center. Hence the set of centers form a dominating set of the graph. Therefore, the size of the

⋆ Supported in part by NSF CCF-0635147.
⋆⋆ This work was done while the author was visiting University of Washington.
⋆⋆⋆ Supported in part by NSF CCF-0343672.
† Supported in part by NSF CCF-0343672.

maximum spanning star forest is the number of vertices minus the size of the minimum dominating set. Computing the maximum spanning star forest of a graph is NP-hard because computing the minimum dominating set is NP-hard.

The spanning star forest problem has found applications in computational biology. Nguyen et al. [9] use the spanning star forest problem to give an algorithm for the problem of aligning multiple genomic sequences, which is a basic bioinformatics task in comparative genomics. The spanning star forest problem and its directed version have found applications in the comparison of phylogenetic trees [3] and the diversity problem in the automobile industry [1].

Surprisingly, even though the maximum spanning star forest is a natural NP-hard problem, there is not much literature on approximation algorithms for this problem. In fact, the first approximation algorithms for this problem appeared recently in the work of Nguyen et al. [9]. They gave a number of approximation algorithms: the most general one being a 0.6-approximation algorithm on an unweighted graph. This should be contrasted with the complementary problem of minimizing the size of the dominating set of the graph which is known to be hard to approximate within a factor of $(1-\epsilon)\ln\, n$ for any $\epsilon > 0$ unless NP is in $DTIME(n^{\log\log n})$ [4, 8]. This disparity in approximability of complementary problems is fairly commonplace (for example the maximum independent set is not approximable to within any polynomial factor while its complement problem of minimum vertex cover can be approximated to within a factor of 2). Nguyen et al. [9] also showed that the spanning star forest problem is hard to approximate to within a factor of $\frac{545}{546}+\epsilon$ unless P=NP. The paper also gave algorithms with better approximation factors for special graphs such as planar graphs and trees (in fact, for trees the optimal spanning star forest can be computed in linear time).

There are some natural weighted generalizations of the spanning star forest problem. The first generalization is when edges have weights and the objective is to maximize the weights of the edges in the spanning star forest solution. There is a simple 0.5-approximation algorithm for this case [9]. Note that the edge-weighted version is no longer the complement of the (weighted) dominating set problem. Another generalization is the case when nodes have weights. The objective now is to maximize the weights of nodes that are leaves in the spanning star forest solution. This problem is the natural complement of the weighted minimum dominating set problem. To the best of our knowledge, the approximability of the node-weighted spanning star forest problem has not been considered before.

### 1.1 Our Results and Techniques

We prove the following results in this paper. First, we improve the result of [9] by giving a 0.71-approximation algorithm for the unweighted spanning star forest problem. Second, we give a 0.64-approximation algorithm for the node-weighted spanning star forest problem. Finally, we prove better hardness of approximation results for the weighted versions of the problem. In particular, we show that the node and edge-weighted spanning star forest problem cannot be approximated to within a factor of $\frac{31}{32}+\epsilon$ and $\frac{19}{20}+\epsilon$, respectively, for any $\epsilon > 0$ unless P=NP.

Our algorithms are based on an LP relaxation of the spanning star forest problem and randomized rounding. For each vertex we have a variable $x_i$ which is 1 if $x_i$ is a leaf. However, the natural rounding scheme of making vertex $i$ a leaf with probability $x_i$ does not give a good approximation ratio. Instead, we make vertex $i$ a leaf with probability $f(t, x_i) = e^{-t(1-x_i)}$, where the value of $t$ is carefully chosen. Note that for fixed $t$, the function $f(t, x_i)$ is non-linear in $x_i$. Non-linear rounding schemes used in ([5, 7]) round with probability $x_i^c$, where $c$ is a fixed constant or is a value that depends on the input[3]. An interesting point about the rounding is that the function $f(t, x_i)$ is nonzero even for $x_i = 0$, so with some low probability, the rounding can round a variable $x_i = 0$ to 1.

The nonlinear rounding algorithm, obtains an approximation factor of $\ln \frac{n}{OPT} + O(1)$ for the dominating set problem, where $n$ is the number of vertices in the graph and $OPT$ is the value of the optimal (fractional) dominating set. This almost matches the best known approximation factor due to Slavík (for the more general set cover problem) [10].

However, the LP rounding only provides a 0.5 approximation, when the dominating set is large (say $0.5n$). To get the claimed factor of 0.71 for unweighted graphs, we use the LP algorithm in conjunction with another algorithm. The idea is to divide the input graph $G$ into the union of a subgraph $G'$ and some trees, where in $G'$ the minimum degree is at least 2. Given a spanning star forest solution for $G'$, we can "lift" back the solution to the original graph $G$. Then we use as a black box the algorithm from [9] that produces a spanning star forest of size at least $\frac{3}{5}n$ on a $n$-vertex graph of minimum degree 2.

We now turn to the node-weighted spanning star forest problem. Our LP rounding algorithm can be easily generalized to the node-weighted case. As in the unweighted case, the LP rounding algorithm by itself does not give us the stated factor of 0.64. To get the claimed approximation factor, we combine our rounding algorithm with the following trivial factor 0.5 algorithm: Compute any spanning tree, designate an arbitrary vertex as root. Divide the tree in to levels based on distance from the root. Make nodes at alternate levels as centers. It is easy to check that one of the two solutions will have weight at least $\frac{1}{2}$ times the sum of the weights of all nodes.

Finally, we turn to our hardness of approximation results. The hardness results are obtained by gadget reductions from the result of Håstad [6] that states that $MAX3SAT$ is NP-hard to approximate to within a factor of $\frac{7}{8}+\epsilon$, for any $\epsilon > 0$, unless P=NP.

[3] For the problem of maximum $k$-densest subgraph, a randomized rounding using $c = 0.5$ appears to be a folklore result that is attributed to Goemans [5].

## 2 Preliminaries

In this paper, we will consider undirected simple graphs that can be unweighted, node-weighted (where weights are on the nodes) or edge-weighted (where the weights are on the edges). Without loss of generality, assume that $G$ is connected, otherwise we can consider each connected components separately. We say a graph is a *star* if there is one vertex (called the *center*) incident to all edges in the graph (all other vertices are called *leaves*). The *size* of a star is the number of edges in the star (for weighted case, it is the sum of weights of the edges or the sum of the weights of the leaves in the star, for edge-weighted and node-weighted stars respectively). In particular, a singleton vertex is a star of size 0.

A *spanning star forest* of a graph $G$ is a collection of node disjoint stars that covers all vertices of $G$. The problem we are interested in is to find a spanning star forest that maximizes the sum of the sizes of its constituent stars. The unweighted, node-weighted and edge-weighted versions of the problem are denoted by UNWEIGHTED SPANNING STAR FOREST, NODE-WEIGHTED SPANNING STAR FOREST and EDGE-WEIGHTED SPANNING STAR FOREST, respectively.

We will now fix some notation. Unless mentioned otherwise, a graph $G = (V, E)$ will be an unweighted graph. For a node-weighted graph, for any vertex $v_i \in V$, its weight will be denoted by $w_i \geq 0$. For an edge-weighted graph, for any edge $e \in E$, its weight will be denoted by $w_e \geq 0$. Further, for a vertex $v_i \in V$, $N(i)$ will denote the neighbor set of $v_i$ in $G$, that is, $N(i) = \{v_j \mid (v_i, v_j) \in E\}$. We will usually denote $|V|$ by $n$. By abuse of notation, we will use $OPT(G)$ to denote the optimal spanning star forest for $G$ as well as its the total size.

Given a maximization problem, we say that an algorithm is an $\alpha$-approximation for $0 < \alpha \leq 1$, if for every input instance the algorithm produces a solution whose objective value is at least $\alpha$ times that of the optimal solution for that instance.

## 3 An LP-Based Algorithm

In this section we will present a linear programming based algorithm for the NODE-WEIGHTED SPANNING STAR FOREST problem. Towards this, we define the following linear programming relaxation. For every vertex $i$, the variable $x_i$ has the following meaning: $x_i = 1$ if $v_i$ is a leaf in the spanning star forest and is 0 otherwise. For a vertex $v_i$, it is not possible to have all vertices in $N(i) \cup \{v_i\}$ as leaves. These constraints have been included in the linear program.

$$\begin{aligned} \max \ & \sum_{v_i \in V} w_i \cdot x_i \\ s.t. \ & x_i + \sum_{v_j \in N(i)} x_j \leq |N(i)|, \quad \forall\, v_i \in V \\ & 0 \leq x_i \leq 1, \quad \forall\, v_i \in V \end{aligned}$$

Let $LP_{OPT}(G)$ be the value of the optimal solution of the LP. For the rest of the section, fix an optimal solution $\{x_i\}_{i \in V}$. Let $W = \sum_{i=1}^{n} w_i$ be the sum of the weights of all the nodes in $G$. Define

$$a = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i x_i}{W}. \tag{1}$$

Notice that this implies that the optimal objective value is $aW$. Note that setting all $x_i = 1/2$ gives a feasible solution with value $W/2$. Thus, $a \geq 1/2$. We will round the given optimal LP solution using the following rounding algorithm.

ROUNDING-ALG.

1. `Make vertex` $v_i$ `a leaf with probability` $e^{-t(a)(1-x_i)}$`,` `where` $t(a) = \frac{1}{a} \ln\left(\frac{1}{1-a}\right)$`. (Note that as` $1/2 \leq a < 1$`,` $t(a) \geq 0$`.)`
2. `Let` $L_1$ `denote the set of vertices declared leaves in the first step.`
3. `Let` $L_2 = \{v_i \in V \mid v_i \cup N(i) \subseteq L_1\}$`. Declare all vertices in` $L_1 \setminus L_2$ `as leaves.`
4. `Assign every leaf vertex to one of its neighbors that is not declared a leaf. Ties are broken arbitrarily.`

We have the following approximation guarantee for the above rounding algorithm.

**Lemma 1.** *Given an LP solution* $\{x_i\}_{i \in V}$, ROUNDING-ALG *outputs a spanning star forest with expected size at least* $aW(1-a)^{\frac{1}{a}-1}$. *That is, it is a* $(1-a)^{\frac{1}{a}-1}$ *factor approximation algorithm for the spanning star forest problem.*

*Proof.* It is easy to verify that ROUNDING-ALG does indeed generate a valid spanning star forest. For notational convenience let $t = t(a)$ where $a$ is as defined in (1). Now the expected total weight of all leaves after step 2 of ROUNDING-ALG is

$$\begin{aligned} \mathbb{E}(\ell_1) &= \sum_{i=1}^{n} w_i e^{-t(1-x_i)} = e^{-t} W \left( \frac{\sum_{i=1}^{n} w_i e^{t x_i}}{W} \right) \\ &\geq e^{-t} W (e^{taW})^{\frac{1}{W}} = W e^{-t(1-a)} \end{aligned}$$

The inequality above is obtained by the fact that the arithmetic mean is larger than the geometric mean, and then using $\sum_{i=1}^{n} w_i x_i = aW$. Now after step 3, a vertex $v_i$ can cease to be a leaf with probability exactly

$$e^{-t(1-x_i)} \prod_{j \in N(i)} e^{-t(1-x_j)}.$$

Thus, if $\ell_2$ is the total weight of vertices that were leaves after step 2 but ceased to be leaves after step 3, then its expectation is given by

$$\begin{aligned}\mathbb{E}(\ell_2) = & \sum_{i=1}^{n} w_i \left( e^{-t(1-x_i)} \prod_{j \in N(i)} e^{-t(1-x_j)} \right) \\ &= \sum_{i=1}^{n} w_i e^{-t} \left( e^{-t(|N(i)| - \sum_{j \in N(i)} x_j - x_i)} \right) \\ &\leq \sum_{i=1}^{n} w_i e^{-t} = W e^{-t}\end{aligned}$$

The inequality follows from the fact that the $x_i$'s form a feasible solution. Now the expected value of the solution produced by ROUNDING-ALG is the expected total weight of leaves at the end of step 3. In other words, the expected value is given by

$$\mathbb{E}(\ell_1) - \mathbb{E}(\ell_2) \geq \; W \left( \frac{e^{at} - 1}{e^t} \right)$$

Now substituting the value $t = \frac{1}{a} \ln \left( \frac{1}{1-a} \right)$ completes the proof. ■

We have the following remarks concerning ROUNDING-ALG.

- The integrality gap of the LP is at most 3/4: consider a 4-cycle. Note that setting all $x_i = 2/3$ is a valid solution, giving an LP optimal value of 8/3. However, the integral optimum value is 2.
- The randomized rounding algorithm can easily be derandomized using the method of conditional expectations [2]. In fact, exact formulas for $\mathbb{E}(\ell_1)$ and $\mathbb{E}(\ell_2)$ are presented in the proof and the conditional expectations are easy to compute from these formulas.
- In the worst case where $a = 1/2$, the approximation ratio of ROUNDING-ALG for spanning star forest is rather bad (equal to 0.5). However, as we will see in the next two sections, we will take advantage of ROUNDING-ALG to get good approximation algorithms.

### 3.1 Application of ROUNDING-ALG to Dominating Set

Observe that the approximation ratio in Lemma 1 improves as the value of $a$ increases. In particular, the approximation ratio tends to 1 as $a$ approaches 1. This suggests that the above rounding scheme yields an approximation algorithm for the complementary objective of minimizing the dominating set. In fact, by analyzing the behavior of the function as $a$ approaches 1, we obtain the following result.

**Theorem 1.** *The* ROUNDING-ALG *computes a* $\left(\ln \frac{W}{OPT_f} + 1 + 2\frac{OPT_f}{W} \ln \frac{W}{OPT_f}\right)$ *approximation ratio solution for the weighted dominating set problem, where* $OPT_f$ *is the total weight of the optimal fractional dominating set solution.*

*Proof.* Let the optimal LP value for the spanning star forest be given by $aW$, where $W$ is the sum of all the node weights. This implies that the optimal (fractional) dominating set has size $OPT_f = (1-a)W$.

Now, the dominating set returned by ROUNDING-ALG has size

$$W - aW(1-a)^{\frac{1}{a}-1} = OPT_f \cdot \frac{1-(1-a)^{\frac{1}{a}-1}a}{1-a}$$

Let $a = 1-\epsilon$. We have

$$\frac{1-(1-a)^{\frac{1}{a}-1}a}{1-a} = \frac{1-\epsilon^{\frac{1}{1-\epsilon}-1}(1-\epsilon)}{\epsilon} = \frac{1-\epsilon^{\frac{\epsilon}{1-\epsilon}}}{\epsilon} + \epsilon^{\frac{\epsilon}{1-\epsilon}}$$

As $\epsilon < 1$, $\epsilon^{\frac{\epsilon}{1-\epsilon}} \leq 1$. Thus, the approximation ratio (for the dominating set problem) is at most:

$$\begin{aligned}\frac{1-\epsilon^{\frac{\epsilon}{1-\epsilon}}}{\epsilon} + 1 &= \frac{1-e^{\frac{\epsilon}{1-\epsilon}\ln\epsilon}}{\epsilon} + 1 \leq \frac{1-\left(1-\frac{\epsilon}{1-\epsilon}\ln\epsilon\right)}{\epsilon} + 1 \\ &= \frac{\frac{\epsilon}{1-\epsilon}\ln\frac{1}{\epsilon}}{\epsilon} + 1 \leq \ln\frac{1}{\epsilon}(1+2\epsilon) + 1 = \ln\frac{1}{\epsilon} + 2\epsilon\ln\frac{1}{\epsilon} + 1,\end{aligned}$$

where in the above we have used that since $0 < \epsilon \leq 1$, $\frac{\epsilon}{1-\epsilon}\ln\frac{1}{\epsilon} < 1$. Further, for any $0 < y < 1$ and $0 < x \leq 1/2$, we have the following inequalities: $e^{-y} \geq 1-y$ and $\frac{1}{1-x} \leq 1+2x$. Note that for our case we can always find a dominating set of size at most $W/2$, that is, $\epsilon \leq 1/2$. The proof is complete by noting that $\epsilon = OPT_f/W$. ■

We remark that $\epsilon = \frac{OPT_f}{W}\ln\frac{W}{OPT_f}$ in general is at most 1. However, if $OPT_f = o(W)$, then $\epsilon = o(1)$. This result is close to the best known bound of $\left(OPT_f - \frac{1}{2}\right)\ln\frac{n}{OPT_f} + OPT_f$ from the analysis of greedy algorithm for set cover (and hence, applicable to dominating set too) in [10].

## 4 An Approximation Algorithm for the UNWEIGHTED SPANNING STAR FOREST Problem

In this section, we will describe a 0.71-approximation algorithm for the UNWEIGHTED SPANNING STAR FOREST problem. We will use the following two known results.

**Theorem 2 ([9]).** *For any connected unweighted graph* $G$ *of minimum degree at least 2, if the number of vertices* $n \geq 8$*, there is a polynomial time algorithm (denoted by* ORACLE-ALG*) to compute a spanning star forest of* $G$ *of size at least* $3n/5$.

**Theorem 3 ([9]).** *For any tree $T$ rooted at $r$, let $OPT_{ct}(T)$ and $OPT_{lf}(T)$ be the optimal value of spanning star forest of $T$ given the condition that $r$ is declared a center and leaf, respectively. Then $OPT_{ct}(T)$ and $OPT_{lf}(T)$ can be computed in polynomial time.*

Starting with the given connected graph $G$, we will generate a subgraph from $G$ recursively as follows: Whenever there is a vertex in the current graph of degree 1, remove the vertex and the edge incident to it from the graph. Denote the final resulting subgraph to be $G'$. Note that $G'$ is connected and every vertex in it has degree at least 2. Let

$$S = \{v_i \in G' \mid \text{at least one edge incident to } v_i \text{ is dropped in the above process}\}.$$

For simplicity, assume $S = \{v_1, \ldots, v_h\}$ and let $(G \setminus G') \cup S$ denote the induced subgraph on the vertex set $(V(G) \setminus V(G')) \cup S$.

Consider the subgraph $(G \setminus G') \cup S$: it is easy to verify that $(G \setminus G') \cup S$ is composed of $h$ disconnected trees rooted at vertices in $S$. Denote these trees by $T_1, \ldots, T_h$, where the root of $T_j$ is $v_j$. Let $OPT_{ct}(T_j)$ and $OPT_{lf}(T_j)$ be the optimal value of spanning star forest for $T_j$ with the condition that $v_j$ is declared a center and leaf, respectively. According to Theorem 3, $OPT_{ct}(T_j)$ and $OPT_{lf}(T_j)$ can be computed in polynomial time. Define

$$S_1 = \{v_j \in S \mid OPT_{ct}(T_j) < OPT_{lf}(T_j)\}$$
$$S_2 = \{v_j \in S \mid OPT_{ct}(T_j) \geq OPT_{lf}(T_j)\}$$

Let $N'(S_2)$ be the set of neighbors of $S_2$ in $G'$. Observe that $|N'(S_2)| \geq 2$ (otherwise, all vertices in $S_2$ would have been removed earlier). Consider the subgraph $G' \setminus S_2$ and assume that there are $k$ vertices in $G' \setminus S_2$. We add two extra vertices $u$ and $v$ and connect $u$ and $v$ to all vertices in $N'(S_2)$. Let the resulting graph be $G^*$ (see Figure 1 for an example). Note that $G^*$ is a connected graph of minimum degree at least 2. Thus by Theorem 2, we can compute a spanning star forest of $G^*$ of size at least $\frac{3}{5} \cdot (k+2)$ in polynomial time.

Now we are ready to describe our algorithm.

TREECUTTING-ALG.

1. For each $i \in S_2$, declare $i$ a center.
2. If the number of vertices in $G' \setminus S_2$ is smaller than (say) $1000$,
3. compute the optimal spanning star forest of $G'$ given vertices in $S_2$ are centers.
4. Else,
5. compute spanning star forests of $G^*$ by ORACLE-ALG and ROUNDING-ALG.
6. declare each $v_i \in G' \setminus S_2$ either a center of leaf according to $\max\{\text{ORACLE-ALG}(G^*), \text{ROUNDING-ALG}(G^*)\}$.
7. Given the choices made for the vertices in $S$, compute the best possible spanning star forest for $T_1, \ldots, T_h$.

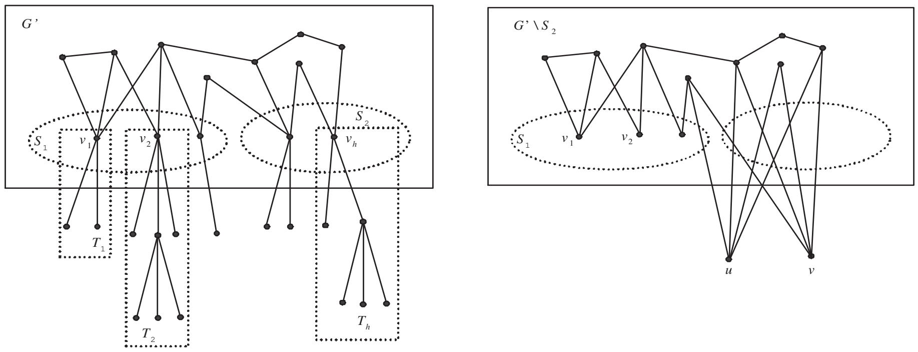


**Fig. 1.** Illustration of Graph $G$ (left) and $G^*$ (right).

Note that all vertices in $S_2$ are declared centers. Thus, in Step 6, the declaration of each vertex $v_i \in G' \setminus S_2$ is feasible (it is either covered by another vertex in $G' \setminus S_2$ or by a vertex in $S_2$). Therefore, the algorithm outputs a feasible spanning star forest solution.

In the following discussions, let $\alpha(G)$ and $\beta(G)$ be the value returned by ORACLE-ALG($G$) and ROUNDING-ALG($G$), respectively. It can be seen that

$$\begin{aligned}&\text{TREECUTTING-ALG}(G)\\ &\geq \max\{\alpha(G^*), \beta(G^*)\} - 2 + \sum_{v_i \in S_1} OPT(T_i \setminus v_i) + \sum_{v_j \in S_2} OPT_{ct}(T_j). \end{aligned} \quad (2)$$

where "$-2$" is because in the worst case, both $u$ and $v$ are leaves in the output of ORACLE-ALG($G^*$) or ROUNDING-ALG($G^*$), but they do not contribute to the solution of $G' \setminus S_2$.

Observe that for any graph $G''$ and any vertex $w \in G''$, given a spanning star forest solution where $w$ is a leaf, we can easily get a solution where $w$ is a center by switching the declaration of $w$ from leaf to center. Thus,

$$OPT(G'' \mid w \text{ is a center}) \geq OPT(G'' \mid w \text{ is a leaf}) - 1.$$

For any $v_j \in S_2$, note that

$$\begin{aligned}&OPT(G' \mid v_j \text{ is a center}) + OPT_{ct}(T_j)\\ &\geq OPT(G' \mid v_j \text{ is a leaf}) - 1 + OPT_{ct}(T_j)\\ &\geq OPT(G' \mid v_j \text{ is a leaf}) - 1 + OPT_{lf}(T_j),\end{aligned}$$

where the second inequality follows from the definition of $S_2$. Therefore,

$$\begin{aligned}OPT(G) = \max\{\ &OPT(G' \mid v_j \text{ is a center}) + OPT_{ct}(T_j),\\ &OPT(G' \mid v_j \text{ is a leaf}) + OPT_{lf}(T_j) - 1\ \}\\ = &OPT(G' \mid v_j \text{ is a center}) + OPT_{ct}(T_j)\end{aligned}$$

In other words, in the optimal solution of $G$, we can always assume vertices in $S_2$ are declared centers.

For any $v_i \in S_1$, we know essentially $OPT_{ct}(T_i) = OPT_{lf}(T_i) - 1$. Note that the root $v_i$ contributes zero to $OPT_{ct}(T_i)$ and one to $OPT_{lf}(T_i)$. That is, regardless of the contribution of $v_i$, the contribution of vertices in $T_i \setminus \{v_i\}$ in $OPT_{ct}(T_i)$ and $OPT_{lf}(T_i)$ is the same. In other words, for any declaration of $v_i$ (either center or leaf), we can always get the same optimal value for $T_i \setminus \{v_i\}$.

Therefore,

$$\begin{aligned} OPT(G) &= OPT(G \mid \text{every } v_j \in S_2 \text{ is a center}) \\ &= OPT(G' \mid \text{every } v_j \in S_2 \text{ is a center}) \\ &\quad + \sum_{v_i \in S_1} OPT(T_i \setminus v_i) + \sum_{v_j \in S_2} OPT_{ct}(T_j). \end{aligned} \tag{3}$$

Thus, when $k$ is small (i.e., TreeCutting-Alg goes through Step 2,3), where recall that $k$ is the number of vertices in $G' \setminus S_2$, TreeCutting-Alg$(G) = OPT(G)$. Hence, we can assume that $k$ is large (i.e., TreeCutting-Alg goes through Step 4,5,6).

Assume that the optimal LP value satisfies $LP_{OPT}(G^*) = a \cdot (k+2)$, where recall that $G^* = (G' \setminus S_2) \cup \{u, v\}$. Hence,

$$\begin{aligned} & \frac{\text{TreeCutting-Alg}(G)}{OPT(G)} \\ \geq\ & \frac{\max\{\alpha(G^*), \beta(G^*)\} - 2 + \sum_{v_i \in S_1} OPT(T_i \setminus v_i) + \sum_{v_j \in S_2} OPT_{ct}(T_j)}{OPT(G' \mid v_j \text{ is a center}, v_j \in S_2) + \sum_{v_i \in S_1} OPT(T_i \setminus v_i) + \sum_{v_j \in S_2} OPT_{ct}(T_j)} && (4) \\ \geq\ & \frac{\max\{\alpha(G^*), \beta(G^*)\} - 2}{OPT(G' \mid v_j \text{ is a center}, v_j \in S_2)} && (5) \\ \geq\ & \frac{\max\{\alpha(G^*), \beta(G^*)\} - 2}{LP_{OPT}(G^*)} && (6) \\ \geq\ & \max\left\{\frac{\frac{3}{5}(k+2)}{a \cdot (k+2)}, \frac{\beta(G^*)}{LP_{OPT}(G^*)}\right\} - \frac{2}{a \cdot (k+2)} && (7) \\ =\ & \max\left\{\frac{0.6}{a}, (1-a)^{\frac{1}{a}-1}\right\} - \frac{2}{a \cdot (k+2)} && (8) \\ >\ & 0.71 && (9) \end{aligned}$$

where (4) follows from (2) and (3), (5) follows from the fact that the summations are non negative, (6) follows from the fact that the LP optimal is larger than the integral optimal value, (7) follows from Theorem 2, (8) follows from Lemma 1, and (9) follows by an estimation using a computer aided numerical analysis (Figure 2).

In conclusion, we have the following result.

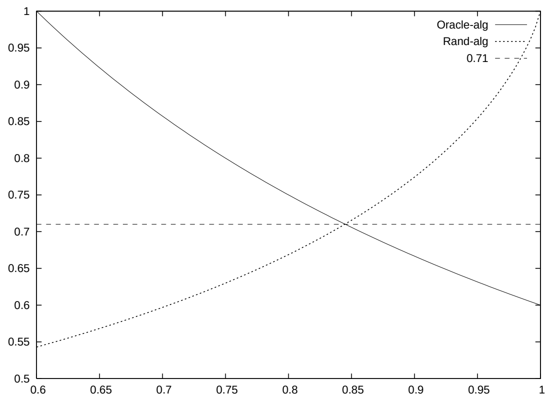


**Fig. 2.** The approximation ratios for ORACLE-ALG and ROUNDING-ALG . The horizontal line is 0.71.

**Theorem 4.** TREECUTTING-ALG *gives a* 0.71*-approximation ratio solution for the* UNWEIGHTED SPANNING STAR FOREST *problem.*

## 5 An Approximation Algorithm for the NODE-WEIGHTED SPANNING STAR FOREST Problem

In this section, we present a 0.64-approximation algorithm for the node-weighted spanning star forest problem. Consider the following simple algorithm.

TRIVIAL-ALG

1. Compute a spanning tree $T$ of the graph $G$, and pick an arbitrary vertex $r$ as its root. Let $h$ denote the height $T$ rooted at $r$. For each integer $k$, let $N_k$ denote the set of vertices at a distance of $k$ (in the tree) from the root $r$.
2. Output the spanning star forest with the higher weight of the following:
   - centers: $N_0 \cup N_2 \cup \ldots$, leaves: $N_1 \cup N_3 \cup \ldots$
   - centers: $N_1 \cup N_3 \cup \ldots$, leaves: $N_0 \cup N_2 \cup \ldots$

   Essentially, the two spanning star forests are obtained by picking alternate levels in the spanning tree $T$.

It is easy to see that the following holds for TRIVIAL-ALG.

**Proposition 1** TRIVIAL-ALG *always outputs a solution with value at least* $W/2$.

**Theorem 5.** *There exists a polynomial time algorithm that solves the* Node-Weighted Spanning Star Forest *problem with an approximation factor of*

$$\min_{a \in [1/2,1)} \max\left(\frac{1}{2a}, (1-a)^{\frac{1}{a}-1}\right) > 0.64$$

*Proof.* Consider the algorithm that runs Trivial-Alg and Rounding-Alg and picks the better of the two solutions– this algorithm obviously has polynomial running time. Let $aW$ denote the value of the LP optimum. From Proposition 1, the Trivial-Alg produces a spanning star forest with weight at least $W/2$, and hence an approximation ratio of at least $\frac{W/2}{aW} = \frac{1}{2a}$. Clearly this also implies that $a > \frac{1}{2}$. The claim on the approximation ratio follows from Lemma 1. The lower bound on the ratio follows by an estimation using a computer aided numerical analysis (Figure 3). ■

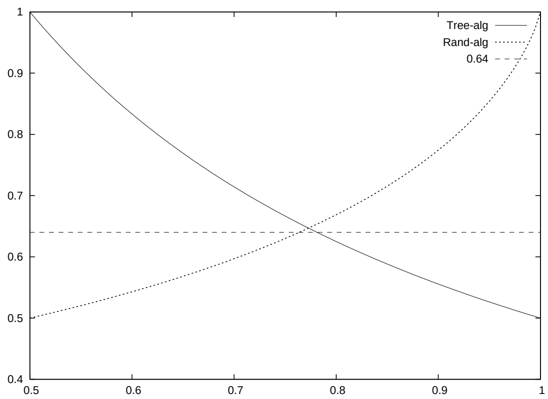


**Fig. 3.** The approximation ratios for Trivial-Alg and Rounding-Alg . The horizontal line is 0.64.

## 6 Hardness of Approximation

The hardness results are obtained by a reduction from the following strong hardness for $MAX3SAT$.

**Theorem 6 ([6]).** *For every $\epsilon > 0$, given a 3-CNF formula $\phi$ it is $NP$-hard to distinguish between the following two cases:*

- *There exists an assignment satisfying $1 - \epsilon$ fraction of the clauses in $\phi$*
- *No assignment satisfies more than $\frac{7}{8} + \epsilon$ fraction of the clauses in $\phi$.*

*Further, the hardness result holds even if each variable $x_i$ is constrained to appear positively and negatively an equal number of times, i.e the literals $x_i, \bar{x_i}$ appear in equal number of clauses.*

**Theorem 7.** *For any $\eta > 0$, it is $NP$-hard to approximate the* Edge-Weighted Spanning Star Forest *problem within $\frac{19}{20} + \eta$.*

*Proof.* Let $\phi$ be a 3-CNF formula on $n$ variables $\{x_1, x_2, \ldots, x_n\}$. Further let $C_1, C_2, \ldots, C_m$ be the set of clauses in $\phi$. From Theorem 6, we can assume that each literal appears positively and negatively an equal number of times. For each $i$, let $d_i$ denote the number of clauses containing $x_i$ (respectively $\bar{x_i}$). Without loss of generality, we assume that $d_i \geq 2$ for all $i$. This can be achieved by just repeating the formula $\phi$ three times. A simple counting argument shows that $\sum_{i=1}^{n} d_i = \frac{3m}{2}$.

Create an edge-weighted graph $G_\phi$ as follows:

- Introduce one vertex $u_i$ for each literal $x_i$ and $v_i$ for literal $\bar{x_i}$, and one vertex $w_j$ for each clause $C_j$. Formally $V = \{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\} \cup \{w_1, \ldots, w_m\}$.
- Introduce an edge between $u_i$ and $w_j$, if clause $C_j$ contains literal $x_i$. Similarly, add an edge $(v_i, w_j)$ if clause $C_j$ contains literal $\bar{x_i}$. Furthermore, for all $i$, introduce an edge between $u_i$ and $v_i$. Formally, $E = \{(u_i, w_j) \mid C_j \text{ contains } x_i\} \cup \{(v_i, w_j) \mid C_j \text{ contains } \bar{x_i}\} \cup \{(u_1, v_1), \ldots, (u_n, v_n)\}$.
- For all $i$, the weight on the edge $(u_i, v_i)$ is equal to $d_i$. The rest of the edges have weight 1.

**Completeness:** Suppose there is an assignment to the variables $\{x_1, \ldots, x_n\}$ that satisfies $1-\epsilon$ fraction of the clauses. Define a spanning star forest as follows:

- Centers : $\{u_i \mid x_i = true\} \cup \{v_i \mid x_i = false\} \cup \{C_j \mid C_j \text{ is not satisfied}\}$.
- Every satisfied clause $C_j$ contains at least one literal which is assigned true. Thus there is a center adjacent to each of the vertices $w_j$ corresponding to a satisfied clause. Since for each $i$, one of $u_i$ or $v_i$ is a center, the other vertex can be a leaf. Thus the set of leaves is given by: $\{u_i \mid x_i = false\} \cup \{v_i \mid x_i = true\} \cup \{w_j \mid C_j \text{ is satisfied}\}$.

Therefore, the total edge weight of the spanning star forest is given by

$$\sum_{i=1}^{n} d_i + |\{w_j \mid C_j \text{ is satisfied}\}| = \sum_{i=1}^{n} d_i + (1-\epsilon)m = \frac{3m}{2} + (1-\epsilon)m = \left(\frac{5}{2} - \epsilon\right) m.$$

**Soundness:** Consider the optimal spanning star forest solution $OPT$ of $G_\phi$. Without loss of generality, we can assume that for each $i$, exactly one of $\{u_i, v_i\}$ is a center, and the other is a leaf attached to it. This is because:

- If both $u_i$ and $v_i$ are centers, then modify the spanning star forest by deleting all the leaves attached to $v_i$, and making $v_i$ a leaf of $u_i$. The total weight of the spanning star forest solution does not decrease, since we delete at most $d_i$ edges of weight 1 and introduce an edge of weight $d_i$.

- If one of $u_i$ and $v_i$ is a center (say $u_i$) and the other (i.e. $v_i$) is a leaf but not attached to $u_i$, then we can disconnect $v_i$ from its center and attach it to $u_i$. This operation increases the weight of the spanning star forest by $d_i - 1$, which contradicts to the optimality of the solution.
- If both $u_i$ and $v_i$ are leaves, then making $u_i$ a center and attaching $v_i$ to it will increase the weight of the solution by $d_i - 2$, again a contradiction.

From the spanning star forest solution $OPT$, obtain an assignment to $\phi$ as follows: $x_i = true$ if $u_i$ is a center in $OPT$ and $x_i = false$ otherwise. If vertex $w_j$ is a leaf in $OPT$, then there is a center (say $u_i$) adjacent to it, which implies that clause $C_j$ is satisfied by the assignment of $x_i$. A similar argument applies when the vertex $w_j$ is adjacent to a center $v_i$. Therefore, the total weight of $OPT$ is given by

$$\sum_{i=1}^{n} d_i + |\{w_j \mid C_j \text{ is satisfied}\}| = \frac{3m}{2} + |\{w_j \mid C_j \text{ is satisfied}\}|$$

In particular, if at most $(\frac{7}{8} + \epsilon)$-fraction of the clauses in $\phi$ can be satisfied, then the weight of $OPT$ is at most $\frac{3m}{2} + (\frac{7}{8} + \epsilon)m = (\frac{19}{8} + \epsilon)m$.

From the completeness and soundness arguments, it is $NP$-hard to distinguish whether $G_\phi$ has a spanning star forest of weight $(\frac{5}{2} - \epsilon)m$ or $(\frac{19}{8} + \epsilon)m$. Thus it is $NP$-hard to approximate the Edge-Weighted Spanning Star Forest problem within a factor of $(\frac{19}{8} + \epsilon)/(\frac{5}{2} - \epsilon)$. The claim follows by picking a small enough $\epsilon$. ■

The proof of the next theorem is similar to the previous one.

**Theorem 8.** *For any $\eta > 0$, it is $NP$-hard to approximate the* Node-Weighted Spanning Star Forest *problem within* $\frac{31}{32} + \eta$.

*Proof.* Let $\phi$ be a 3-CNF formula on $n$ variables $\{x_1, x_2, \ldots, x_n\}$ and $m$ clauses $C_1, C_2, \ldots, C_m$. From Theorem 6, we can assume that each literal appears positively and negatively an equal number of times. For each $i$, let $d_i$ denote the number of clauses containing $x_i$ (respectively $\bar{x_i}$).

Create a node-weighted graph $G_\phi$ as follows:

- Introduce three vertices $a_i, u_i, v_i$ for each variable $x_i$, and one vertex $w_j$ for each clause $C_j$. Formally $V = \{a_1, \ldots, a_n\} \cup \{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\} \cup \{w_1, \ldots, w_m\}$.
- Introduce an edge between $u_i$ and $w_j$, if clause $C_j$ contains literal $x_i$. Similarly, add an edge $(v_i, w_j)$ if clause $C_j$ contains the literal $\bar{x_i}$. Furthermore, for all $i$, introduce edges $(a_i, u_i), (u_i, v_i), (v_i, a_i)$. Formally, $E = \{(u_i, w_j) \mid C_j$ contains $x_i\} \cup \{(v_i, w_j) \mid C_j$ contains $\bar{x_i}\} \cup \{(a_1, u_1), (u_1, v_1), (v_1, a_1), \ldots, (a_n, u_n), (u_n, v_n), (v_n, a_n)\}$
- For all $i$, the weight of nodes $a_i, u_i, v_i$ is equal to $d_i$. The weight of the rest of nodes is 1.

**Completeness:** Suppose there is an assignment to the variables $\{x_1, \ldots, x_n\}$ that satisfies $1-\epsilon$ fraction of the clauses. Define a spanning star forest solution as follows:

- Centers : $\{u_i \mid x_i = true\} \cup \{v_i \mid x_i = false\} \cup \{C_j \mid C_j \text{ is not satisfied}\}$.
- Every satisfied clause $C_j$ contains at least one literal which is assigned true. Thus there is a center adjacent to each of the vertex $w_j$ corresponding to a satisfied clause. Since for each $i$, one of $u_i$ or $v_i$ is a center, the other remaining two in $\{a_i, u_i, v_i\}$ can be leaves. Thus the set of leaves is given by : $\{u_i \mid x_i = false\} \cup \{v_i \mid x_i = true\} \cup \{w_j \mid C_j \text{ is satisfied}\} \cup \{a_i\}$.

The total node weight of the spanning star forest solution is given by

$$\sum_{i=1}^{n} 2d_i + |\{w_j \mid C_j \text{ is satisfied}\}| = \sum_{i=1}^{n} 2d_i + (1-\epsilon)m = 3m + (1-\epsilon)m = (4-\epsilon)\,m.$$

The rest of the proof is similar to that of Theorem 7 and is omitted due to space considerations. ■

## References


1. A. Agra, D. Cardoso, O. Cerfeira, and E. Rocha. A spanning star forest model for the diversity problem in automobile industry. In *ECCO XVIII, Minsk*, May 2005.
2. N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley and Sons, Inc., 1992.
3. V. Berry, S. Guillemot, F. Nicholas, and C. Paul. On the approximation of computing evolutionary trees. In *Proceedings of the Eleventh Annual International Computing and Combinatorics Conference*, pages 115–123, 2005.
4. U. Fiege. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
5. M. X. Goemans. Mathematical programming and approximation algorithms. September 1996. Lecture at Udine School, Udine, Italy.
6. J. Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.
7. R. Krauthgamer, A. Mehta, and A. Rudra. Pricing commodities, or how to sell when buyers have restricted valuations. 2007. Manuscript.
8. C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.
9. 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. In *SODA*, pages 645–654, 2007.
10. P. Slavík. A tight analysis of the greedy algorithm for set cover. In *STOC*, pages 435–441, 1996.