

Approximating the Spanning Star Forest Problem and Its Applications to Genomic Sequence Alignment

C. Thach Nguyen* Jian Shen† Minmei Hou‡ Li Sheng§ Webb Miller¶
Louxin Zhang||

Abstract

This paper studies the algorithmic issues of the spanning star forest problem. We prove the following results: (1) there is a polynomial-time approximation scheme for planar unweighted graphs; (2) there is a polynomial-time algorithm with approximation ratio $\frac{3}{5}$ for unweighted graphs; (3) it is NP-hard to approximate the problem within ratio $\frac{545}{546} + \epsilon$ for unweighted graphs; (4) there is a linear-time algorithm to compute the maximum star forest of a weighted tree; (5) there is a polynomial-time algorithm with approximation ratio $\frac{1}{2}$ for weighted graphs. We also show how to apply this spanning star forest model to aligning multiple genomic sequences over a tandem duplication region.

1 Introduction

A star is a graph in which some vertex is incident to each of the edges of the graph. A star forest is a graph in which each component is a star. Star forests have previously appeared in the literature on star arboricity [2]. The star arboricity of a graph G is the minimum number of star forests whose union contains all edges of G . Bounds on star arboricity have been established for several classes of graphs including planar graphs [3, 4, 10, 13].

A spanning star forest of a graph is a spanning subgraph of G in which each component is a star. The size of a spanning star forest SF_G of a graph G is defined to be the number of edges of SF_G if G is unweighted and the total weight of all edges of SF_G if G is weighted.

Even in an unweighted graph, spanning star forests may have different sizes. As in any forest, the size of a spanning star forest of an unweighted graph G is equal to the number of vertices of G minus the number of stars in the star forest.

Although the size of a spanning star forest of an unweighted graph and its relationship to the dominating number are observed [12], the problem of finding a maximum spanning star forest of a graph has not been well studied yet. Our work focuses on this algorithmic problem.

Our motivation for studying this problem comes from aligning multiple genomic sequences, a basic bioinformatics task in comparative genomics. When a set of genomic sequences over a tandemly duplicated region are aligned, the number of alignment blocks output from the current existing programs such as TBA often explodes exponentially [15]. To control the size of output alignment blocks and guarantee the alignment quality simultaneously, we propose to define a so-called alignment graph using the pairwise similarity of the given sequences and use a maximum spanning star forest as a guide for building alignment blocks (see Section 5 for details). In addition, the spanning star forest problem and its directed version also arise from comparison of phylogenetic trees [6] and the diversity problem in automobile industry [1].

Each spanning star forest of an unweighted graph G corresponds to a dominating set of G , since the set of centers of the stars in a spanning star forest is a dominating set of G . Because of this, the problem of finding a maximum spanning star forest is proved to be NP-hard in general, but to be polynomial-time solvable for trees and to have a polynomial-time approximation scheme for planar unweighted graphs (in Section 3.1). In Section 3.2, we also present a polynomial-time algorithm of approximation ratio $\frac{3}{5}$ for unweighted graphs. However, we prove that it is NP-hard to approximate the problem within ratio $\frac{545}{546} + \epsilon$ for unweighted graphs (in Section 3.3) for any small $\epsilon > 0$.

For weighted graphs, the spanning star forest problem is not equivalent to the smallest dominating set

*Department of Mathematics, National University of S'pore, S'pore 117543. Email: matnct@nus.edu.sg

†Department of Mathematics, Texas State University, San Marcos, TX 78666, USA. Email: js48@txstate.edu

‡Department of Comput. Sci. & Eng., The Penn State University, University Park, PA 16802, USA. Email: mhoul@bx.psu.edu

§Department of Math, Drexel University, Philadelphia, PA 19104. Email: lsheng@math.drexel.edu

¶Department of Biology and Department of Comput. Sci. & Eng., The Penn State University, University Park, PA 16802, USA. Email: webb@bx.psu.edu

||Department of Mathematics, National University of S'pore, S'pore 117543. Email: matzlx@nus.edu.sg

problem. In Section 4, we present a dynamic programming algorithm for weighted trees; we also give a simple polynomial-time algorithm of approximation ratio $\frac{1}{2}$ for arbitrary graphs.

Finally, in Section 5, we show how to apply this spanning star forest model to aligning multiple genomic sequences over a tandemly duplicated region.

2 The spanning star forest problem

In this paper, we consider undirected, weighted or unweighted, simple graphs. A *star* is a graph having a vertex (called the center) incident to each of all edges in the graph. The center of a star S has the maximum degree. If a star has only two vertices, either vertex can be its center. A *star forest* is a graph in which each component is a star. The *size* of a graph G is the number of edges in G if G is unweighted; it is the sum of the edge weights if G is weighted. A *spanning star forest* of a graph G is a star forest that contains all the vertices of G . Different spanning star forests of G may have different sizes. In the rest of paper, we study the following algorithmic problem:

Spanning Star Forest

INSTANCE: A (unweighted or weighted) graph $G = (V, E)$.

OBJECTIVE: Find a spanning star forest of G that has the largest size.

As we shall prove in next section, this problem is NP-hard. Hence, we shall focus on developing approximation algorithms for it. An approximation algorithm for the spanning star forest problem has approximation ratio $r < 1$ if it outputs a spanning star forests of size at least $r \cdot OPT_{sf}(G)$ for any graph G , where $OPT_{sf}(G)$ is the maximum size of a spanning star forest of G .

3 Algorithms for unweighted graphs

3.1 Spanning star forest and dominating set

The dominating set is one of the most important concepts in graph theory [14]. Given a (unweighted) graph $G = (V, E)$, a subset of vertices $D \subseteq V$ is called a *dominating set* if, for every $v \in V$, there is at least one vertex $u \in D$ that is adjacent to it, i.e., $(u, v) \in E$. Assume $D = \{v_1, v_2, \dots, v_k\}$ is a k -vertex dominating set of G . For each $u \in V - D$, we select a unique vertex $v_u \in D$ such that $(u, v_u) \in E$ and associate the edge (u, v_u) to u . For each $d \in D$, all the associated edges of the form (v, d) give rise to a star S_d centered at d . Trivially, $\cup_{d \in D} S_d$ is a spanning star forest of G . As in any spanning forest of d components, there are $n - d$ edges in $\cup_{d \in D} S_d$, where n is the number of vertices in G . Note that from a dominating set, one may construct different spanning star forests with the same size.

Conversely, given a spanning star forest F of size k of G , the centers of the stars in F form a dominating set that contains $n - k$ vertices. In summary, we have the following simple fact.

LEMMA 3.1. *Let the largest size of a spanning star forest of a graph G be denoted by $\alpha(G)$ and the dominating number of G be denoted by $\gamma(G)$. Then, $\alpha(G) = n - \gamma(G)$, where n is the order of G .*

This implies that finding a maximum spanning star forest of an unweighted graph is equivalent to finding a minimum dominating set. Therefore, as the dominating set problem [9], the spanning star forest problem can be solved in linear time for trees. The lemma also implies the following result.

THEOREM 3.1. *The spanning star forest problem has a polynomial-time approximation scheme (PTAS) for planar graphs.*

Proof. It is known that the dominating set problem has a PTAS for planar graphs [5]. For any small $\epsilon > 0$, there is an approximation algorithm \mathcal{A}_ϵ that produces a $(1+\epsilon)$ -approximation solution in polynomial-time in terms of the length of the input for fixed ϵ . For any connected planar graph G with n vertices, its dominating number $\gamma(G)$ satisfies $\gamma(G) \leq \frac{n}{2}$. We first obtain a dominating set D_ϵ of at most $(1 + \epsilon)\gamma(G)$ vertices by applying \mathcal{A}_ϵ to G . Then, we construct a spanning star forest F_ϵ from D_ϵ as described before Lemma 3.1. Since $\gamma(G) \leq \frac{n}{2}$, by Lemma 3.1, F_ϵ has size

$$n - |D_\epsilon| \geq n - (1 + \epsilon)\gamma(G) \geq (1 - \epsilon)(n - \gamma(G)) = (1 - \epsilon)\alpha(G).$$

Thus, F_ϵ is a $(1 - \epsilon)$ -approximation solution of the spanning star forest problem.

3.2 A $\frac{3}{5}$ -approximation algorithm

THEOREM 3.2. *If there is a polynomial-time algorithm that finds a dominating set of at most $(\frac{1}{2} - \epsilon)n$ vertices given an n -vertex graph of minimum degree 2, then there is a polynomial-time ratio- $(\frac{1}{2} + \epsilon)$ approximation algorithm for the spanning star forest problem.*

Proof. Let \mathcal{A} be the algorithm that outputs a dominating set of at most $(\frac{1}{2} - \epsilon)n$ vertices given an n -vertex graph of minimum degree 2. Consider a graph G . A vertex is called a support vertex if it is adjacent to some degree-1 vertices. Without loss of generality, we assume G contains l degree-1 vertices and an even number of support vertices u_1, u_2, \dots, u_{2k} , $k \geq 0$. Obviously, $l \geq 2k$. We first construct a graph H from G by

removing all the degree-1 vertices and adding k vertices v_1, v_2, \dots, v_k and the following $2k$ edges

$$(u_{2i-1}, v_i), (u_{2i}, v_i), 1 \leq i \leq k.$$

Then, H has $n - l + k$ vertices of minimum degree 2. Applying \mathcal{A} to H , we obtain a dominating set D_H of at most $(\frac{1}{2} - \epsilon)(n - l + k)$ vertices. Now we construct a dominating set D_G of G as follows: for each i , if $v_i \in D_H$, we remove v_i from D_H and add u_{2i-1} and u_{2i} into the set; if $v_i \notin D_H$, then at least one of u_{2i-1} and u_{2i} is in D_H , and we add the other into the set. In other words, $D_G = (D_H \setminus \{v_1, v_2, \dots, v_k\}) \cup \{u_1, u_2, \dots, u_{2k}\}$. It is easy to verify that D_G is a dominating set. By the way of constructing D_G , its size is at most

$$|D_H| + k \leq (\frac{1}{2} - \epsilon)(n - l + k) + k \leq (\frac{1}{2} - \epsilon)n + k$$

since $l \geq 2k$. This implies that a spanning star forest F of G can be obtained from D_G with at least

$$\begin{aligned} n - |D_G| &\geq n - [(\frac{1}{2} - \epsilon)n + k] \\ &= (\frac{1}{2} + \epsilon)n - \frac{1}{2}(2k) \\ &= \frac{1}{2}(n - 2k) + \epsilon n \end{aligned}$$

edges. Since any dominating set of G must contain each support vertex or its degree-1 neighbors, $\gamma(G) \geq 2k$, and hence $\alpha(G) \leq n - 2k$. Therefore, F contains at least $(\frac{1}{2} + \epsilon)\alpha(G)$ edges. This has proved that \mathcal{A} can be extended into a ratio- $(\frac{1}{2} + \epsilon)$ approximation algorithm for the spanning star forest problem.

THEOREM 3.3. *There is a $\frac{3}{5}$ -approximation algorithm for the spanning star forest problem.*

Proof. McCuaig and Shepherd proved that any n -vertex graph of minimum degree 2 has a dominating set of size at most $\frac{2}{5}n$ for any $n \geq 8$ (see [17]). We demonstrate that such a dominating set can be found in polynomial-time (see Appendix A). Hence, by Theorem 3.2, there is a polynomial-time algorithm of approximation ratio $\frac{1}{2} + \frac{1}{10} = \frac{3}{5}$ for the problem.

3.3 Hardness of approximation We have shown that the spanning star forest problem can be approximated within a constant ratio in polynomial-time. On the other hand, the dominating set problem cannot be approximated within $(1 - \epsilon)\ln n$ for any ϵ unless $NP \subset DTIME(n^{\log \log n})$ [16, 11]. Hence, intuitively, the spanning star forest problem should not be approximated with some constant ratio in polynomial-time. Now, we prove this fact rigorously.

THEOREM 3.4. *The spanning star forest problem cannot be approximated within a ratio $\frac{545}{546} + \epsilon$ in polynomial-time for any $\epsilon > 0$ unless $P = NP$.*

Proof. We prove this fact using an L -reduction from the VERTEX COVER problem. Recall that the VERTEX COVER problem is to find the smallest subset $U \subset V$ such that every edge has at least one endpoint in U given a graph $G = (V, E)$. It is well known that this problem cannot be approximated within a ratio $79/78 - \epsilon$ for graphs with the maximum degree at most 4 unless $P = NP$ [7]. Given a graph $G = (V_G, E_G)$ in which the maximum degree is at most 4, we construct a graph $H = (V_H, E_H)$ from G by adding a length-2 path parallel to each edge in G . Formally,

$$\begin{aligned} V_H &= V_G \cup \{v_e \mid e \in E_G\}, \\ E_H &= E_G \cup \{(x, v_e), (v_e, y) \mid e = (x, y) \in E_G\}. \end{aligned}$$

It is easy to see that the following facts hold:

For any $V \subset V_G$, if it is a vertex cover set of G , then it is a dominating set of H . Conversely, for each subset $V \in V_H$, if it is a dominating set of H , then $(V \cap V_G) \cup \{x \in V_G \mid v_e \in V \& e = (x, y) \in E\} \subseteq V_G$ is a vertex cover of H .

Let $Opt_{vc}(G)$, $Opt_d(H)$ and $Opt_{sf}(H)$ denote the minimum size of a vertex cover of G , the minimum size of a dominating set of H , and the maximum size of a spanning star forest of H respectively. Then, the above facts imply that $Opt_d(H) = Opt_{vc}(G)$ and

$$\begin{aligned} &Opt_{sf}(H) \\ &= |V_H| - Opt_d(H) = (|V_G| + |E_G|) - Opt_d(H) \\ &= (|V_G| + |E_G|) - Opt_{vc}(G) \leq 7Opt_{vc}(G) \end{aligned}$$

since $|V_G| \leq |E_G| \leq 4Opt_{vc}(G)$, since the degree of each vertex is at most 4 in G . Assume S is a spanning star forest of H . If S contains k stars, then, the size $|S|$ of S is $|V_H| - k$. In addition, these k centers of S induce a vertex cover V_S of size at most k of G . Hence, since $Opt_d(H) = Opt_{vc}(G)$,

$$\begin{aligned} &|V_S| - Opt_{vc}(G) \\ &\leq k - Opt_{vc}(G) = k - Opt_d(H) = Opt_{sf}(H) - |S|. \end{aligned}$$

Thus, if $|S|$ is a ratio- $(1 - \frac{1}{7} \cdot \frac{1}{78} + \epsilon)$ approximation of the spanning star forest problem for H , where $\epsilon > 0$, i.e., $|S| \geq (1 - \frac{1}{7} \cdot \frac{1}{78} + \epsilon)Opt_{sf}(H)$, then the resulting vertex cover set V_S satisfies

$$\begin{aligned} &|V_S| - Opt_{vc}(G) \\ &\leq (\frac{1}{7} \cdot \frac{1}{78} - \epsilon)Opt_{sf}(H) \leq (\frac{1}{78} - 7\epsilon)Opt_{vc}(G), \end{aligned}$$

by the first inequality shown in this proof. In other words, V_S is a ratio- $(\frac{79}{78} - 7\epsilon)$ approximation of the VERTEX COVER problem for G . This implies that the spanning star forest problem cannot be approximated within a ratio $1 - \frac{1}{7} \cdot \frac{1}{78} + \epsilon = \frac{545}{546} + \epsilon$ in polynomial-time unless $P = NP$.

Figure 1: (a) A weighted graph; (b) The unique maximum spanning star forest with an isolated vertex.

4 Algorithms for weighted graphs

4.1 A remark on maximum spanning star forests A spanning star forest with the largest size may contain a star without edges, i.e. an isolated vertex. However, in a connected unweighted graph, we can always find a maximum spanning star forest without isolated vertices. Let G be a connected unweighted graph. Consider a maximum spanning star forest \mathcal{F} in which there is an isolated vertex, say u . Since G is connected, u is incident to some edge (u, v) of G . If v is a center of a star in \mathcal{F} (e.g., v is an isolated vertex, or v is a vertex of a two-vertex star), then, we could obtain a larger spanning star forest by merging u into the star centered at v , a contradiction. If v is a leaf of a star with at least 3 vertices, then we delete the edge between v and the center, and put an edge between u and v to form star $\{u, v\}$. Repeating this procedure, we will eventually derive a maximum spanning star forest without isolated vertices.

In a connected weighted graph, every maximum spanning star forest may contain some isolated vertices. For example, the graph shown in Figure 1 has a unique spanning star forest with the maximum weight 7. Therefore, when we design an algorithm for the spanning star forest problem for connected weighted graph, we have to consider the star forests with isolated vertices.

4.2 A linear-time algorithm for weighted trees Trees are the simplest connected graphs. In this subsection, we present a linear-time algorithm for finding the maximum spanning star forest of a tree.

Given a tree T , we first root T at an arbitrary vertex r and consider each edge (u, v) as a directed edge from the endpoint closer to the root to the other endpoint. In the rest of this subsection, when we mention that (u, v) is an edge, we mean that u is closer to the root; we say that u is the parent of v or v is a child of u if (u, v) is an edge in the rooted tree. Obviously, in a star forest of T , the root can be a center of a star, a leaf of a star centered at one of its children, or an isolated vertex. For each vertex u of T , we let $T(u)$ be the subtree rooted at u and define the following three numbers:

$\Phi(u)$: The maximum weight of a spanning star forest of $T(u)$ in which u is a center of a multiple-vertex star;

$\Psi(u)$: The maximum weight of a spanning star forest of $T(u)$ in which u is a leaf of a multiple-

vertex star;

$\Omega(u)$: The maximum weight of a spanning star forest of $T(u)$ in which u is an isolated node.

First, these three numbers can be computed through recurrence formulas as shown below.

LEMMA 4.1. *Let $C(u)$ be the set of the children of u . Then,*

$$\begin{aligned}\Phi(u) &= \sum_{v \in C(u)} \Delta(v) \\ &\quad - \min_{v \in C(u)} (\Delta(v) - \Omega(v) - w(uv)) \\ \Psi(u) &= \max_{v \in C(u)} [w(uv) + \max\{\Phi(v), \Omega(v)\}] \\ &\quad + \sum_{x \in C(u) - \{v\}} \max\{\Phi(x), \Psi(x), \Omega(x)\} \\ \Omega(u) &= \sum_{v \in C(u)} \max\{\Phi(v), \Psi(v), \Omega(v)\}\end{aligned}$$

where $\Delta(v) = \max\{\Phi(v), \Psi(v), \Omega(v) + w(uv)\}$.

Proof. For any vertex u of T , let $SFR(u)$ be a star forest of $T(u)$ that has a maximum weight $\Phi(u)$, over all the star forests in which u is a center of a star S_u . For any $v \in C(u)$, we use $SFR(u)|_v$ to denote the restriction of $SFR(u)$ in the subtree $T(v)$ rooted at v and define $\Delta(v) = \max\{\Phi(v), \Psi(v), \Omega(v) + w(uv)\}$.

We consider the following cases.

Case 1 The star S_u centered at u contains at least two leaves.

If v is a leaf of the star S_u centered at u in $SFR(u)$, then $SFR(u)|_v$ must be a star forest of $T(v)$ that has the maximum weight $\Omega(v)$, over all the star forests in which v is an isolated vertex. Moreover, $w(uv) + \Omega(v) \geq \Phi(v), \Psi(v)$. Equivalently, $\Delta(v) = w(uv) + \Omega(v)$. Otherwise, we could obtain a star forest with larger weight from $SFR(u)$ by removing edge uv and replacing $SFR(u)|_v$ by a star forest (of $T(v)$) in which v is not isolated, contradicting our assumption that $SFR(u)$ has the maximum weight.

If v is not a leaf of the star S_u , then $SFR(u)|_v$ is a star forest (of $T(v)$) with weight $\Delta(v)$, which is $\max\{\Psi(v), \Phi(v)\}$. Otherwise, we can obtain a star forest of $T(u)$ with a larger weight by replacing $SFR(u)|_v$ by another star forest of weight $\Omega(v)$ and adding the edge uv into $T(u)$.

This implies that $\Phi(u) = \sum_{v \in C(u)} \Delta(v) = \sum_{v \in C(u)} \max\{\Phi(v), \Psi(v), \Omega(v) + w(uv)\}$. For any $v \in C(u)$, by definition, $\Delta(v) - w(uv) - \Omega(v) \geq 0$. For any v that is a leaf of the star S_u , by the above argument, $\Delta(v) = w(uv) + \Omega(v)$. Therefore, in this case, $\min_{v \in C(u)} (\Delta(v) - w(uv) - \Omega(v)) = 0$ and hence,

$$\begin{aligned}\Phi(u) &= \sum_{v \in C(u)} \max\{\Phi(v), \Psi(v), \Omega(v) + w(uv)\} \\ &\quad - \min_{v \in C(u)} (\Delta(v) - w(uv) - \Omega(v)).\end{aligned}$$

Case 2 The star S_u centered at u contains only one leaf.

Let v' be the unique leaf in S_u . Then, for any $v \in C(u)$, we have $\Delta(v) - w(uv) - \Omega(v) \geq \Delta(v') - w(uv') - \Omega(v')$. Otherwise, we could obtain a star forest of a larger weight from $SFR(u)$ by the following operations:

- (a) Add edge uv ,
- (b) Delete uv' ,
- (c) Replace $SFR(u)|_v$ by a star forest (of $T(v)$) in which v is isolate, and
- (d) Replace $SFR(u)|_{v'}$ by the star forest (of $T(v')$) with the maximum weight $\Delta(v')$.

Hence, the weight $\Phi(u)$ of the star forest $SFR(u)$ is equal to

$$\begin{aligned} & \sum_{v \in C(u) - \{v'\}} \Delta(v) + (w(uv') + \Omega(v')) \\ = & \sum_{v \in C(u)} \Delta(v) - (\Delta(v') - w(uv') - \Omega(v')) \\ = & \sum_{v \in C(u)} \Delta(v) - \min_{v \in C(u)} (\Delta(v) - w(uv) - \Omega(v)). \end{aligned}$$

This proves the recurrence formula for $\Phi(u)$

Let SFL be a star forest (of $T(u)$) in which u is a leaf of a star S_u centered at one of its children, $v \in C(u)$. Then, $SRL|_v$ is a star forest (of $T(v)$) in which v is isolated or the center of a star. Hence, $SRL|_v$ has weight $\max\{\Phi(v), \Omega(v)\}$. For any other $v' \neq v$ in $C(u)$, $SFL|_{v'}$ is a star forest (of $T(v')$) in which v' can be a center of a star, a leaf of a star, or an isolated vertex, and so $SFL|_{v'}$ has weight $\Delta(v')$. Hence,

$$\begin{aligned} \Psi(u) = & \max_{v \in C(u)} [w(uv) + \max\{\Phi(v), \Omega(v)\}] \\ & + \sum_{x \in C(u) - \{v\}} \max\{\Phi(x), \Psi(x), \Omega(x)\}. \end{aligned}$$

This proves the recurrence formula for $\Psi(u)$.

Let $SFI(u)$ be the star forest (of $T(u)$) with the maximum weight $\Omega(u)$ in which u is isolated. Then, For each $v \in C(u)$, $SFI(u)|_v$ has to be a star forest of $T(v)$ with the maximum weight $\Delta(v)$. Otherwise, $SFI(u)$ does not have the maximum weight $\Omega(u)$. Therefore, the recurrence formula for $\Omega(u)$ is correct. This finishes the proof of the lemma.

THEOREM 4.1. *There is a linear-time algorithm that outputs a star forest with the maximum weight given a weighted tree.*

Proof. Lemma 4.1 implies a dynamic programming approach for computing the maximum star forest of a weight tree rooted at r . For each u in the tree, compute $\Psi(u), \Phi(u), \Omega(u)$ from the corresponding values at its children according to the recurrence formulas given in the lemma in linear-time.

After $\Phi(r), \Psi(r), \Omega(r)$ are computed, the maximum star forest can be found by backtracking along the computation path. We call a star forest $SF(u)$ in which u is a center of a star, a leaf of star, or an isolated vertex as a type-1, type-2 or type-3 star forest. The goal of finding the maximum star forest can be achieved by introducing backtracking pointers: r-ptr, l-ptr, i-ptr, which are used to construct the maximum star forest of type-1, type-2 and type-3 at each vertex. At each vertex u , r-ptr(u) specifies (a) which child of u is in the star S_u centered at u , and (b) if $v \in C(u)$ is not in the star S_u , the type of the restriction star forest on $T(v)$; l-ptr(u) specifies (a) which child of u is the center of the star containing u , and (b) the type of the restriction star forest on $T(v)$ for each $v \in C(u)$; i-ptr(u) specifies the type of the restriction star forest on $T(v)$ for each $v \in C(u)$. Obviously, with these backtracking pointers, the maximum star forest can be found in linear time. This finishes the proof.

4.3 A $\frac{1}{2}$ -approximation algorithm For an arbitrary weighted graph, we find a spanning star forest using the following algorithm:

1. Find a maximum spanning tree T_G of the given graph G ;
2. Compute a maximum spanning star forest SF_G of T_G

Given a positively-weighted connected graph, its maximum spanning tree can be computed in polynomial-time. For example, we can use Kruskal's algorithm for the purpose. Since Step 2 also takes polynomial-time, the above method can be executed in polynomial-time.

Let $OPT_{sf}(G)$ and $OPT_t(G)$ be the maximum weight of a spanning star forest and a spanning tree of G respectively. Since G is connected, any spanning star forest can be extended into a spanning tree by adding some 'bridge' edges between the forests. Hence, $OPT_{sf}(G) \leq OPT_t(G)$. In addition, since any tree can be decomposed into two edge-disjoint star forests (see, for example, [2, 4]),

$$w(SF_G) \geq \frac{1}{2} w(T_G) = OPT_t(G) \geq OPT_{sf}(G).$$

Therefore, the above algorithm has approximation ratio $1/2$. This has proved the following theorem.

THEOREM 4.2. *The spanning star forest problem can be approximated within ratio $\frac{1}{2}$ in polynomial-time for arbitrary weighted graphs.*

5 Application to aligning genomic sequences

To align multiple genomic sequences over different species, the TBA program [8] works on the phylogenetic tree T over the species in a bottom-up fashion. At each internal node v of T , the program outputs a set of multiple alignments of segments (called blocks) in the given sequences by merging the blocks generated at the left and right children of v through pairwise alignments between sequences contained in left and right children. The program stops at the root of T , outputting a set of blocks (called a blockset). The output blockset can be considered as a packing of the pairwise alignments between sequence segments generated at each internal nodes.

Genomic sequences of human, mouse and other higher organisms are full of tandem repeats. In these genomes, a gene family may have dozens or even hundreds of members. Assume we align a set of whole genomes. Consider a gene family that has multiple copies in each species. At an internal node v of T , in the worst case, an alignment program will generate a pairwise alignment between every pair of the orthologous gene sequences contained in the left and right subtrees respectively. As a result, every pair of blocks that contain the orthologous gene sequences and that are generated at the left and right children of v respectively will be merged into a larger block. Hence, the number of blocks that contain the gene sequences is equal to the product of the numbers of blocks generated in the left and right children of v . When the program stops at the root of T , it will output a blockset of an exponentially large size.

To avoid the number of blocks growing exponentially, we can generate just a linear number of blocks at each internal node by carefully selecting the blocks to be merged. More specifically, for the node v , we use B_v to denote the blockset generated at v and $|B_v|$ its size, i.e. the number of blocks contained in B_v . Obviously, when v is a leaf in T , $B_v = \phi$. Let v' and v'' be the left and right children of v respectively. A size explosion can be avoided by arranging that the blockset B_v contains at most $|B_{v'}| + |B_{v''}| + c$ blocks, where c is a constant independent of $|B_{v'}|$ and $|B_{v''}|$. In this way, for a gene family having g_i copies in species i of N species, the final output blockset will contain at most $\sum_{i=1}^N g_i + cN$ blocks over the gene family.

Let P be the set of pairwise alignments generated at v . We denote an alignment in P with rows a and b by $a.b$, where a and b are segments of sequences in the left and right subtree respectively. Define

$$L_P = \{a \mid a.b \in P \text{ for some } b\}$$

and

$$R_P = \{b \mid a.b \in P \text{ for some } a\}.$$

Our method selects blocks in $B_{v'}$ and $B_{v''}$ to merge according to the following theory.

We first define a weighted bipartite graph G_v with vertex bipartition (V', V'') . The vertices in V' correspond one-to-one to the blocks in $B_{v'}$ and the rows in L_P that are not contained in any blocks in $B_{v'}$; the vertices in V'' correspond to the blocks in $B_{v''}$ and the rows in R_P in the same way. For each $x \in V'$, $y \in V''$, there is an edge between x and y if and only if there is a pairwise alignment $a.b \in P$ such that a and b appear in the blocks corresponding to x and y respectively, called a *reference alignment* of the blocks. In general, there are multiple reference alignments for each pair of blocks. Hence, the weight of the edge (x, y) is then defined to be the maximum alignment score over all the reference alignments of the blocks corresponding to x and y . In practice, the rows in P can partially overlap with some rows of a block in $B_{v'} \cup B_{v''}$. Here, however, we assume a row in P is either contained in or disjoint from any row in $B_{v'} \cup B_{v''}$. Since each genomic sequence contains many different orthologous sequences, the resulting bipartite graph G_v has more than one connected components in general. By construction, none of the connected components in the graph are singletons.

To control the size of the output blockset, we make use of a maximum spanning star forest of G_v to merge the blocks $B_{v'}$ and $B_{v''}$ as shown in the *Block-Merging Algorithm*. One desired property of a spanning star forest SF of a graph G is that each edge has at least one degree-1 endpoint in SF . Such a property is critical to controlling the size of the output blockset as shown below.

BLOCK-MERGING ALGORITHM

Input: $B_{v'}$, $B_{v''}$ and P .

0. $B_v = \phi$;
1. Construct the graph G_v by using $B_{v'}$, $B_{v''}$ and P ;
2. Heuristically compute a maximum spanning star forest SF of G_v ;
3. For each edge (x, y) in SF , merge the blocks correspondent to x and y , add the resulting block into B_v ;
4. Output B_v .

THEOREM 5.1. *The BLOCK-MERGING ALGORITHM outputs a blockset satisfying the following properties:*

Full coverage property *Any position covered by a pairwise alignment produced during*

the aligning process appears in some block in the output blockset.

Non-redundancy property *Each block in the output blockset contains a unique row that does not appear in any other blocks.*

Proof. The full coverage property is obviously. We prove the non-redundancy property by induction on the depth of an internal node. Let v be an internal node. If the children v' and v'' of v are leaves, then $B_{v'}$ and $B_{v''}$ are empty and hence, each vertex of the bipartite graph G_v corresponds to a segment in some given sequence. For each edge $e = (x, y)$ in the spanning star forest of G_v computed in Step 3 of the algorithm, one of x and y , say x , is of degree 1 and hence only incident to e . Then, the segment corresponding to x is only covered by the block derived from e through merging the two segments corresponding to x and y respectively. Therefore, each block in B_v has a unique row.

If at least one of v' and v'' are not a leaf, by induction, we assume that both $B_{v'}$ and $B_{v''}$ satisfy the non-redundancy property if they are not empty. Consider a block $Z \in B_v$ obtained through an edge (x, y) of the spanning star forest found in Step 3 of the algorithm. Without loss of generality, we assume that x is of degree 1 in the spanning star forest. If x corresponds to a block in $B_{v'}$, then the unique row in the block corresponding to x only appears in Z among all the blocks in B_v . If x corresponds to a row R_x of a pairwise alignment in P , then, by the definition of G_v , R_x is not contained in any block in $B_{v'}$ and $B_{v''}$ and hence appears uniquely in Z among all the blocks in B_v . This concludes the proof.

Theorem 5.1 can be used to estimate the base-pair size $\|B\|$ of the blockset B output from the algorithm. Let N be the number of sequences in the input genomic sequence set S . Assume that any pair of orthologous sequence segments s and s' have roughly equal length, i.e. $|s| = \Theta(|s'|)$. For each block $X \in B$, we use $r_{unique}(X)$ to denote the unique row in X . By assumption, the base-pair size $|X|$ of X is at most $N\Theta(|r_{unique}(X)|)$ since X has at most N rows. Therefore, using the fact that $r_{unique}(X)$'s are disjoint segments of the input sequences in S , we have

$$\|B\| = \sum_{X \in B} |X| \leq \sum_{X \in B} N\Theta(|r_{unique}(X)|) \leq N\Theta\left(\sum_{s \in S} |s|\right).$$

In other words, the basis-pair size of the output blockset is at most N times the base-pair size of the input sequence set.

6 Conclusions

In this paper, we initiate the algorithmic study of the spanning star forest problem. In particular, we give a $\frac{3}{5}$ -approximation algorithm for unweighted graphs and a $\frac{1}{2}$ -approximation algorithm for weighted graphs. In contrast, we also prove that it is NP-hard to approximate the problem within a ratio $\frac{545}{546} + \epsilon$. The study raises several research questions for future study. For example, how to obtain an approximation algorithm with better ratio and how to improve the inapproximability result is how to design good algorithms for the spanning star forest problem for directed, weighted graphs.

7 Acknowledgment

CT Nguyen and LX Zhang were supported by NUS ARF grant R-146-000-068-112. MM Hou and W Miller were supported by NIH grant HG002238. L Sheng was supported by NSF grant CCR-0311413.

References

- [1] A. Agra et al., *A spanning star forest model for the diversity problem in automobile industry*, ECCO XVIII, Minsk, Belarus, May 26-28, 2005.
- [2] J. Akiyama and M. Kano, *Path factors of a graph*, in Graph Theory and its Applications (eds: F Haray and J Maybee), pp. 1-21, Wiley, New York, 1984.
- [3] Y. Aoki, *The star-arboricity of the complete regular multipartite graphs*, Discrete Math 81(1990), pp. 115-122.
- [4] I. Algor and N. Alon, *The star arboricity of graphs*. Discrete Math 75 (1989), pp. 11-22.
- [5] B. S. Baker, *Approximation algorithms for NP-complete problems on planar graphs*, J. of Assoc. for Comput. Machinery 41 (1994), pp. 153-180.
- [6] V. Berry, S. Guillelot, F. Nicolas, and C. Paul, *On the approximation of computing evolutionary trees*, Proc. of the 11th Annual Inter'l Computing and Combinatorics Conference, LNCS, vol. 3595, pp. 115-125, Springer-Verlag, 2005.
- [7] P. Berman and M. Karpinski, *On some tighter inapproximability results*, Proc. ICALP 1999, LNCS vol. 1644, pp. 200-209.
- [8] M. Blanchette et al. *Aligning multiple genomic sequences with the threaded blockset aligner*, Genome Res. 14(2004), pp. 708-715.
- [9] E. J. Cockayne, S. E. Goodman, S. T. Hedetniemi, *A linear algorithm for the domination number of a tree*, Information Processing Letters 4 (1975), pp. 41-44.
- [10] Y. Egawa, T. Fukada, S. Nagoya, and M. Urabe, *A decomposition of complete bipartite graphs into edge-disjoint subgraphs with star components*, Discrete Math 58(1986), pp. 93-95.

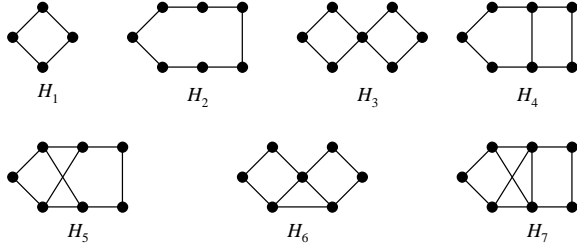


Figure 2: Seven bad graphs whose minimum dominating sets have size larger than $2n/5$ (reproduced from [17]).

- [11] U. Feige, *A threshold of $\ln n$ for approximating set cover*, Journal of the ACM, 45(1998), pp. 634–652. Also appeared in Proc. STOC’96.
- [12] S. Ferneyhough et al., *Star forests, dominating sets and Ramsey-type problems*, Discrete Math 245 (2002), pp. 255-262.
- [13] S. I. Hakimi, J. Mitchem, and E. Schmeichel, *Star arboricity of graphs*, Discrete Math 149 (1996), pp. 93-98.
- [14] T. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.
- [15] M. M. Hou, P. Berman, L. X. Zhang and W. Miller, *Controlling size when aligning multiple genomic sequences with duplications*, to Appear in Proc of WABI 2006.
- [16] C. Lund and M. Yannakakis, *On the hardness of approximation minimization problems*, Journal of ACM 41 (1994), pp. 961-981.
- [17] W. McCuaig and B. Shepherd, *Domination in graphs with minimum degree two*, J. of Graph Theory 13 (1989), pp. 749-762.

Appendix A: Finding a Dominating Set of Size at most $2n/5$

THEOREM 7.1. *Let $G = (V, E)$ be a connected graph such that each vertex has degree at least 2. If G is not one of the 7 graphs in Figure ?? (called bad graphs) then $\gamma(G) \leq 2/5|V|$. Moreover, such a dominating set can be found in $O(|E|^2|V|)$ time.*

Proof. We use $\delta(G)$ to denote the minimum degree of a vertex in G . We prove by induction on $|V|$. It is easy to check that for $|V| \leq 7$, we can always find a dominating set satisfying the theorem. Now assume that $|V| > 7$ and that the theorem holds for all graphs G' whose number of vertices is smaller than $|V|$. We show how to construct a dominating set of G whose size is at most $2/5|V|$.

If G is not edge-minimal with respect to connectedness and minimum degree of 2, we keep deleting edges. Therefore, in the following, we assume that G is edge-

minimal with respect to these conditions.

Let $B(G) = \{u \in V | d(u) > 2\}$. If $|B(G)| = 0$ then G is a cycle. In this case, a dominating set of G can be found by numbering its vertices starting from 0 and choosing all the vertices whose indices is divisible by 3. It is readily verified that this dominating set contains at most $2/5|V|$ vertices.

If $|B(G)| = 1$ then G is a collection of cycles having a unique common vertex u . Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be the set of cycles of length 4, 7 and other lengths respectively. Furthermore, let $\mathcal{C} = \{C_1, C_2, \dots, C_t\}$. A dominating set of G can be constructed by the following steps: (i) construct a dominating set of size 2 that contains u for each cycle in \mathcal{A} ; (ii) construct a dominating set of size 3 that contains u for each cycle in \mathcal{B} ; (iii) construct a dominating set of size at most $2/5|C_i|$ for each i ; (iv) union all the dominating sets. The size of this dominating set is

$$1 + |\mathcal{A}| + 2|\mathcal{B}| + \sum (2/5n_i - 1) = 1 + |\mathcal{A}| + 2|\mathcal{B}| + 2/5 \sum n_i - |\mathcal{C}|.$$

which is larger than $2/5|V|$ only if G is either H_1 H_2 or H_3 .

Now assume that $|B(G)| > 2$. In the following, for convenience, we will denote V and E as the vertex and edge set of a graph G without mention. We consider the following cases

Case 1: There is an edge uv between two vertices in $B(G)$. Due to the minimality of G , this edge must be a bridge of G . Let G_u and G_v be the components of $G - uv$ containing u and v respectively. Then both G_u and G_v are connected and $\delta(G_u) \geq 2$ and $\delta(G_v) \geq 2$.

- Case 1.1: both G_u and G_v are good. We can find dominating sets D_u and D_v of G_u and G_v such that $|D_u| \leq 2/5|V_u|$ and $|D_v| \leq 2/5|V_v|$. $D_u \cup D_v$ is a dominating set of G whose size is at most $2/5|V|$.
- Case 1.2: only, say, G_v is a good graph. Let G_u^* be the graph where $V_u^* = V_u \cup \{v\}$ and $E_u^* = E_u \cup \{uv\}$ and G_v^* be the graph constructed by: (i) choosing a neighbor v' of v in G_v ; (ii) joining v' with all other neighbors of v in G_v and (iii) deleting v . Then, G_v^* is connected and $\delta(G_v^*) \geq 2$ while G_u^* contains one of the graphs in Figure ?. Note that H_4, H_5, H_7 in Figure ? contain H_2 and H_6 contains H_3 and so we only need to consider the cases in Figure ?. From this figure, we can see that G_u^* has a dominating set D_u^* that contains v where $|D_u^*| \leq 2/5|V_u^*|$. Consider the goodness of G_v^* .

On one hand, if G_v^* is good, it has a dominating set D_v^* of size at most $2/5|V_v^*|$. Then $D_u^* \cup D_v^*$ is a dominating set of G whose size is at most $2/5|V|$.

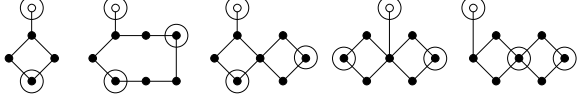


Figure 3: A graph formed by adding a vertex to a bad graph contains one of these graphs. The white vertex is the additional vertex. The circles mark the vertices in a dominating set of each graph.

On the other hand, if G_v^* is bad then G_v contains one of the graphs in Figure ??, thereby having a dominating set D_v containing v such that $|D_v| \leq 2/5|V_v|$. Let $D = D_u^* \cup D_v$, then $|D| = |D_u^*| + |D_v| - 1$ and D is a dominating set of G . Clearly, $|D| \leq 2/5(|V_u^*| + |V_v| - 1) = 2/5|V|$.

- Case 1.3: Both G_u and G_v are bad. Consider the graphs G_u^* such that $V_u^* = V_u \cup \{v\}$ and $E_u^* = E_u \cup \{uv\}$ and $G_v^* = G_v - \{v\}$. Again, G_u^* has a dominating set of size at most $2/5|V_u^*|$ and G_v^* has a dominating set of size at most $2/5|V_v^*|$. Then the union of these two dominating set gives a dominating set of G whose size is at most $2/5|V|$.

Case 2: There is no edge between any two vertices in $B(G)$. In the following, we will use *2-cycles* to refer to the cycles of G which contains exactly one vertex in $B(G)$ and *2-paths* to refer to the paths of G whose end points are in $B(G)$ but none of the internal points is. The vertex of degree larger than 2 on a 2-cycle is called its *end point*.

If G has a 2-cycle of length at least 6 or a 2-path of length at least 5 then there is a path $vv_1v_2v_3v'$ in G such that $d(v_1) = d(v_2) = d(v_3) = 2$ and v' is neither v nor joined to v . A graph G' is formed by removing v_1, v_2, v_3 from and adding the edge vv' to G . Clearly G' is connected and $\delta(G') \geq 2$. Since $|V| > 7$, G' is not H_1 . If G' is bad then $G' - v$ has a dominating set D' of size 2; $D' \cup \{v, v_2\}$ is a dominating set of G with size 4, which is $2/5|V|$. If G' is good then it has a dominating set D' of size at most $2/5|V'|$. We construct a dominating set D of D' as follow: (i) if $v \in D'$ then $D = D' \cup \{v_3\}$; (ii) if $v' \in D'$ then $D = D' \cup \{v_1\}$ and (iii) if neither v nor v' is in D' then $D = D' \cup \{v_2\}$. Then $|D| = |D'| + 1 \leq 2/5(|V'| + 3) = 2/5|V|$.

Now assume that all 2-cycles in G are of length at most 5 and all 2-paths in G are of length at most 4.

- Case 2.1: G contains a 2-cycle of length 4 whose end point's degree is larger than 3. Let this cycle be $vv_1v_2v_3v$ where v is the end point. A graph G' is formed by deleting v_1, v_2 and v_3 from v . Again, G' is not H_1 and $\delta(G') \geq 2$ since the degree of v

is larger than 3 in G . If G' is bad, $G' - v$ has a dominating set D' of size 2. Then $D' \cup \{v, v_2\}$ is a dominating set of G whose size is $4 = 2/5|V|$. If G' is good, it has a dominating set D' of size at most $2/5|V'|$. Then $D' \cup \{v_2\}$ is a dominating set of G whose size is smaller than $2/5(|V'| + 3) = 2/5|V|$.

- Case 2.2: every 2-cycle of length 4 in G has an end point of degree 3. Consider such a cycle $vv_1v_2v_3v$ where v is the end point. The other neighbor v' of v has degree 2, since there is no edge between two vertices in $B(G)$. The subgraph of G induced by $\{v, v_1, v_2, v_3, v'\}$ is called a 4-cluster and v' is called the *anchor* of this cluster.

If G contains a 4-cluster, say $C = \{v, v_1, v_2, v_3, v'\}$, with anchor v' such that the v 's neighbor u that is not in C has degree larger than 2 in G , then G' is a connected graph and $\delta(G') \geq 2$. If G' is bad, similar to above, $G' \cup \{uv'\}$ has a dominating set D' of size at most $2/5(|V'| + 1)$ which contains v' . Then $D' \cup \{v_2\}$ is a dominating set of G whose size is less than $2/5(|V'| + 4) = 2/5|V|$. On the other hand, if G' is good, then it has a dominating set D' such that $|D'| \leq 2|V'|/5$. Hence, $D = D' \cup \{v, v_2\}$ is a dominating set of G and $|D| \leq 2|V|/5$.

Now assume the otherwise, we further define each 2-cycle of length 5 or 3 in G to be a 5-cluster or 3-cluster respectively and the anchors of these clusters are the end points of the corresponding cycles. Finally, each vertex u such that $d(u) > 2$ and u is not an end point of any 2-cycles forms a 1-cluster with u its anchor.

If there is an edge joining two anchors, the two anchors must be that of two 4-clusters and there is only one graph admitting this. A dominating set of size 4 of this graph, which has 10 vertices, can easily be found.

If there is no edge joining two anchors, a dominating set D of G is formed by collecting all anchors and one vertex in each 4-cluster or 5-cluster. We will prove that $|D| \leq 2/5|V|$. Let m_5, m_4, m_3, m_1 be the number of 5-clusters, 4-clusters, 3-clusters and 1-clusters respectively. The number of anchors of 4-clusters and 1-clusters are m_4 and m_1 respectively. However, an anchor can be shared between a number of 5-clusters and 3-clusters; therefore, if we let m_h be the number of vertices which are anchors of some 3-clusters or 5-clusters then $m_h \leq m_5 + m_3$. Clearly, $|D| = m_h + m_5 + 2m_4 + m_1$.

Now we count the non-anchor vertices. Each 5-cluster, 4-cluster, 3-cluster and 1-cluster contains 4, 4, 2 and 0 non-anchor vertices respectively. Fur-

thermore, on each 2-path connecting two clusters, there are 1 or 2 non-anchor vertices; therefore, the number of non-anchor vertices on these paths is at least $(m_4 + m_h + 3m_1)/2$. Hence, the number of non-anchor vertices is at least $4m_5 + 9m_4/2 + 2m_3 + 3m_1/2 + m_h/2$. This implies $|V| \geq 4m_5 + 11m_4/2 + 2m_3 + 3m_h/2 + 5m_1/2$. Simple computation shows that $|D| \leq 2/5|V|$.

The theorem follows from the above cases.

A recursive algorithm to compute such a dominating set follows from the above induction proof. The algorithm finds a dominating set iteratively. Each iteration takes $O(|E|^2)$ time and removes some vertices from the graph. Therefore, the running time of the algorithm is $O(|E|^2|V|)$.