

APPROXIMATING THE SPANNING STAR FOREST PROBLEM AND ITS APPLICATION TO GENOMIC SEQUENCE ALIGNMENT*

C. THACH NGUYEN[†], JIAN SHEN[‡], MINMEI HOU[§], LI SHENG[¶], WEBB MILLER^{||}, AND
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 graphs; (2) there is a polynomial-time $\frac{3}{5}$ -approximation algorithm for graphs; (3) it is NP-hard to approximate the problem within ratio $\frac{259}{260} + \epsilon$ for graphs; (4) there is a linear-time algorithm to compute the maximum star forest of a weighted tree; (5) there is a polynomial-time $\frac{1}{2}$ -approximation algorithm for weighted graphs. We also show how to apply this spanning star forest model to aligning multiple genomic sequences over a tandem duplication region.

Key words. dominating set, spanning star forest, approximation algorithm, genomic sequence alignment

AMS subject classifications. 68Q17, 68Q25, 68R10, 68W25

DOI. 10.1137/070682150

1. Introduction. A star is a tree in which some vertex is incident to each of the edges in the graph. Let G be a graph. A spanning star forest SF_G of a graph G is a spanning subgraph of G in which each connected component is a star. The size of SF_G is defined to be the number of edges in SF_G ; if G is weighted, the size of SF_G is defined to be the sum of the weights of all edges in SF_G . Even in a graph, spanning star forests may have different sizes. As in any forest, the size of a spanning star forest of a graph G is equal to the number of vertices of G minus the number of stars in the star forest. We call the vertex that is incident to all edges the center of a star. A subset of vertices dominates G if every other vertex is adjacent to at least one vertex in the subset. It is not hard to see that the centers of the stars in a spanning star forest form a dominating set for G . Hence, finding a maximum spanning star forest of a graph is equivalent to finding a smallest dominating set. The latter is a fundamental problem in algorithmic graph theory.

Although the size of a spanning star forest of a graph and its relationship to the dominating number have been observed [13], the problem of finding a maximum

*Received by the editors February 8, 2007; accepted for publication (in revised form) February 25, 2008; published electronically June 6, 2008. The extended abstract of this work appeared in the Proceedings of the 18th Annual SIAM–ACM Symposium on Discrete Algorithms (SODA’07).

<http://www.siam.org/journals/sicomp/38-3/68215.html>

[†]Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350 (ncthach@cs.washington.edu). This work was done when this author was at the National University of Singapore. This author was supported by NUS ARF grant R-146-000-068-112.

[‡]Department of Mathematics, Texas State University, San Marcos, TX 78666 (js48@txstate.edu).

[§]Department of Computer Science, Northern Illinois University, DeKalb, IL 60115 (mhou@cs.niu.edu). This work was done when this author studied at Penn State University. This author was supported by NIH grant HG002238.

[¶]Department of Mathematics, Drexel University, Philadelphia, PA 19104 (lsheng@math.drexel.edu). This author was supported by NSF grant CCR-0311413.

^{||}Department of Biology and Department of Comput. Sci. & Eng., Penn State University, University Park, PA 16802 (webb@bx.psu.edu). This author was supported by NIH grant HG002238.

^{**}Department of Mathematics, National University of Singapore, Singapore 117543 (matzlx@nus.edu.sg). This author was supported by NUS ARF grant R-146-000-068-112.

spanning star forest of a graph has yet to be well studied. 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. A speciation is an evolutionary process that creates new species. Two genes from different species are orthologous if they diverged as the result of a speciation event. A duplication is an evolutionary event in which a genomic segment is copied and inserted at a different position. Two related genes could have diverged due to an intraspecies duplication event. To perform comparative genome analysis, it is desirable to produce a multi-species orthologous alignment, in which there is at most one row of sequence from a given species in each alignment block. The many-to-many orthologous relationships among duplicated genes causes a dramatic explosion of the alignment size when the multispecies alignment contains all combinations of pairwise orthologous relationships [16].

Currently, there are no good solutions to aligning duplication-rich genomic regions. The existing threaded blockset aligner (TBA) program screens out duplicated alignments and thus is not able to capture all pairwise orthologous relationships in a duplicated-gene cluster [7]. For example, using TBA, each human alpha-globin gene is aligned to only one rat alpha-globin gene, despite the fact that a human alpha-globin gene is actually orthologous to several rat alpha-globin genes, and those other alignments are lost. To control the size of the computed alignment blocks while guaranteeing the alignment quality, we propose to define a so-called alignment graph using the pairwise similarity of the given sequences and then utilize a maximum spanning star forest of the resulting alignment graph as a guide for building alignment blocks (see section 5 for details).

In addition, the spanning star forest problem has potential application in comparison of phylogenetic trees [6]. A directed star is defined to be outward if all arcs are from the center to a leaf. The directed version of the spanning star forest problem is, given a directed graph, to find a maximum spanning subgraph in which each connected component is an outward star. Such a directed version of the spanning star forest problem arises from the diversity problem in the automobile industry [1].

The dominating set problem is a well-known NP-hard problem. Because of this, the problem of finding a maximum spanning star forest is NP-hard in general but is polynomial-time solvable for trees and has a polynomial-time approximation scheme (PTAS) for planar graphs, as indicated in section 3.1. In section 3.2, we also present a polynomial-time algorithm of approximation ratio $\frac{3}{5}$ for any graph. On the other hand, we prove in section 3.3 that it is NP-hard to approximate the problem within ratio $\frac{259}{260} + \epsilon$ for any small $\epsilon > 0$.

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

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

2. The spanning star forest problem. In this paper, we consider simple graphs that are undirected and connected, weighted or unweighted. We simply say that G is a graph if it is unweighted. A *star* is a tree 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.

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, 11, 14].

The *size* of a graph G is the number of edges in G , and 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 the 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 always outputs a spanning star forest of size at least $r \cdot \text{OPT}_{sf}(G)$ given a graph G , where $\text{OPT}_{sf}(G)$ is the maximum size of a spanning star forest of G . We call such an algorithm an r -approximation algorithm.

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 [15]. Given a graph $G = (V, E)$, a subset of vertices $D \subseteq V$ is called a *dominating set* if, for every $v \in V - D$, 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 v_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 domination 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 a graph is equivalent to finding a minimum dominating set. Therefore, as the dominating set problem [10], the spanning star forest problem can be solved in linear time for trees. The lemma also implies the following result.

THEOREM 3.2. *The spanning star forest problem has a PTAS for planar graphs.*

Proof. Recall that $\gamma(G)$ denotes the dominating number of a graph G . Since the dominating set problem has a PTAS for planar graphs [5], for any small $\epsilon > 0$, there is a polynomial-time algorithm \mathcal{A}_ϵ of approximation ratio $(1 + \epsilon)$ for the problem. 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 $\gamma(G) \leq \frac{n}{2}$ (Ore theorem; see [15], for example), 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 for G . \square

3.2. A $\frac{3}{5}$ -approximation algorithm. In this subsection, we mainly present a polynomial-time $\frac{3}{5}$ -approximation algorithm using a known upper bound on the size of a dominating set in a graph.

THEOREM 3.3. *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 at least 2, then there is a polynomial-time $(\frac{1}{2} + \epsilon)$ -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. Suppose G contains l degree-1 vertices. For simplicity, we first assume that G contains 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

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

Then, H has $n - l + k$ vertices of degree at least 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, $D_G = (D_H - \{v_1, v_2, \dots, v_k\}) \cup \{u_1, u_2, \dots, u_{2k}\}$. It is easy to verify that D_G is a dominating set. By the construction of D_G , its size is at most

$$(3.1) \quad |D_H| + k \leq \left(\frac{1}{2} - \epsilon\right)(n - l + k) + k \leq \left(\frac{1}{2} - \epsilon\right)n + \left(\frac{1}{2} + \epsilon\right)k$$

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

$$(3.2) \quad n - |D_G| \geq n - \left[\left(\frac{1}{2} - \epsilon\right)n + \left(\frac{1}{2} + \epsilon\right)k\right] > \left(\frac{1}{2} + \epsilon\right)(n - 2k)$$

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.

When G contains an odd number of support vertices $u_1, u_2, \dots, u_{2k+1}$, $k \geq 1$, we modify the construction of H presented above by adding two new vertices x_{k+1} and x_{k+2} and three edges (u_{2k+1}, x_{k+1}) , (x_{k+1}, x_{k+2}) , (x_{k+2}, u_{2k+1}) . Now, H has $n - l + k + 2$ vertices. We derive dominating set D_G of G from the output dominating set D_H from the algorithm as

$$D_G = (D_H - \{v_1, v_2, \dots, v_k, x_{k+1}, x_{k+2}\}) \cup \{u_1, u_2, \dots, u_{2k+1}\}.$$

The size of D_G is at most $|D_H| + k$ and the inequalities (3.1) and (3.2) become

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

and

$$n - |D_G| \geq n - \left[\left(\frac{1}{2} - \epsilon \right) (n + 1) + \left(\frac{1}{2} + \epsilon \right) k \right] > \left(\frac{1}{2} + \epsilon \right) (n - 2k - 1),$$

respectively. Since $\alpha(G) \leq n - 2k - 1$, we have that the corresponding spanning star forest F contains at least $(\frac{1}{2} + \epsilon)\alpha(G)$ edges. Hence, the theorem also holds in this case. \square

THEOREM 3.4. *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 [18]). We demonstrate that such a dominating set can be found in polynomial time by giving a different constructive proof of their theorem (see Appendix A). Hence, by Theorem 3.3, there is a polynomial-time algorithm of approximation ratio $\frac{1}{2} + \frac{1}{10} = \frac{3}{5}$ for the problem. \square

The above result could be improved by using a better upper bound on $\gamma(G)$ for graphs G with large minimum degree $\delta(G)$. If $\delta(G) \geq 7$, $\gamma(G) \leq |V_G|[1 - \delta(G)(\frac{1}{\delta(G)+1})^{1+1/\delta(G)}]$ (Theorem 2.8 in [15]). Although such a bound is not true for $\delta(G) = 3$, $\gamma(G) \leq \frac{3}{8}|V_G|$ if $\delta(G) \geq 3$ (see [19]). If there is a polynomial-time algorithm that, given a graph of minimum degree at least 3, always outputs a dominating set of at most $\alpha|V_G|$, $\frac{3}{8} \leq \alpha < \frac{2}{5}$, the following theorem implies a better approximation algorithm.

THEOREM 3.5. *Let $\frac{3}{8} \leq \alpha < \frac{2}{5}$. If there is a polynomial-time algorithm that finds a dominating set of at most αn vertices given an n -vertex graph of minimum degree at least 3, then there is a polynomial-time $(\frac{4}{5} - \frac{1}{2}\alpha)$ -approximation algorithm for the spanning star forest problem.*

Proof. Let $G = (V, E)$ be a graph with n vertices, where $n \geq 8$. We denote the subset of vertices of degree i by V_i , and hence $V = \cup_i V_i$. First, we construct a maximal subset $A_1 \subseteq V_1$ such that the distance between any two vertices in A_1 is at least 3. Recall that a vertex is called a support vertex if it is adjacent to some degree-1 vertices. It is easy to see that each support vertex is adjacent to a unique vertex in A_1 . We next construct an $A_2 \subseteq V_2$ having the following property:

- (i) Any pair of vertices in $A_1 \cup A_2$ has distance at least 3; and
- (ii) A_2 is maximal; i.e., $A_2 \cup \{u\}$ does not satisfy condition (i) for any $u \in V_2 - A_2$.

Such a subset can be constructed recursively in polynomial time. Initially, we set $A_2 = \phi$ and add the vertices in V_2 one by one to A_2 subject to condition (i).

Let a_j denote the number of vertices in A_j , $j = 1, 2$. By condition (i), no vertex in G can dominate more than one vertex in $A_1 \cup A_2$. Thus,

$$(3.3) \quad \gamma(G) \geq a_1 + a_2.$$

For each vertex u , we use $N(u)$ to denote the set of its neighbors. Define $B_j = \cup_{u \in A_j} N(u)$, $j = 1, 2$. By the above observation, B_1 is identical to the set of the support vertices and $|B_1| = |A_1| = a_1$. Hence, for each $u \in V_1 - A_1$, there is a unique $v \in A_1$ such that the distance $d(u, v)$ between u and v is 2.

Since no two vertices in A_2 have a common neighbor, $|B_2| = 2a_2$. By the maximality property of A_2 , for each $u \in V_2 - A_2$, there is some $v \in A_1 \cup A_2$ such that $d(u, v) \leq 2$. If $d(u, v) = 1$, then $u \in B_1 \cup B_2$. If $d(u, v) = 2$, then u is adjacent to some vertex in $B_1 \cup B_2$. This implies that the disjoint union $B_1 \cup B_2$ dominates $V_1 \cup V_2$.

Now, we construct from G a graph G' such that $\delta(G') \geq 3$ as follows.

Step 1. Delete all the degree-1 vertices. For simplicity, we assume that 3 divides $|B_1|$. Partition B_1 into groups of 3 vertices each. For each group of vertices x, y, z , we add a new vertex u_{xyz} and three edges $u_{xyz}x, u_{xyz}y$, and $u_{xyz}z$. We also add edges xy, yz , and zx if any of them is not in G . In this way, x, y, z, u_{xyz} all have degree at least 3. After the completion of step 1, the number of vertices in the resulting graph is

$$n - |V_1| + \frac{|B_1|}{3} \leq n - |B_1| + \frac{|B_1|}{3} = n - \frac{2a_1}{3}.$$

Step 2. Add edge uv if it is not in G for every $u, v \in V_2 - A_2$. This makes each vertex in $V_2 - A$ have degree at least three if $|V_2 - A_2| \geq 4$. (In case $0 \leq |V_2 - A_2| \leq 3$, in the argument below $D \cup B_1 \cup V_2$ would be a dominating set of G , and so the right-hand side of inequality (3.4) would be replaced by $\alpha n + \frac{2}{3}(1 - \alpha)a_1 + \frac{1}{3}(1 - \alpha)a_2 + 3$.)

Step 3. Delete all vertices in A_2 . For simplicity, again, we assume that 3 divides $|B_2|$. Partition B_2 into groups of 3 vertices each. For each group of three vertices x, y, z , add a new vertex u_{xyz} and three edges $u_{xyz}x, u_{xyz}y$, and $u_{xyz}z$. We also add edges xy, yz , and zx if any of them is not in G . As a result, x, y, z, u_{xyz} all have degree at least 3. After this step is done, the resulting graph G' has at most

$$n - \frac{2a_1}{3} - a_2 + \frac{|B_2|}{3} = n - \frac{2a_1}{3} - \frac{a_2}{3}$$

vertices and $\delta(G') \geq 3$.

By applying the polynomial-time algorithm to G' , we obtain a dominating set D' of size at most $(n - \frac{2a_1}{3} - \frac{a_2}{3})\alpha$. Since $B_1 \cup B_2$ dominates $V_1 \cup V_2$, $D' \cup B_1 \cup B_2$ induces a dominating set D_1 for G : For each group x, y, z in steps 1 and 3, if u_{xyz} is in D' , we replace it with x, y, z . The resulting dominating set D_1 is of size at most

$$(3.4) \quad |D'| + \frac{2(|B_1| + |B_2|)}{3} \leq \alpha n + \frac{2(1 - \alpha)}{3}a_1 + \frac{4 - \alpha}{3}a_2.$$

Notice that B_1 is the set of support vertices and $|B_1| = a_1$. Applying inequality (3.1) with the McCuaig–Shepherd bound ($\frac{1}{2} - \epsilon = \frac{2}{5}, k = \frac{a_1}{2}$), we can also obtain a dominating set D_2 of size at most $\frac{2}{5}n + \frac{3}{10}a_1$. By selecting the smaller one between D_1 and D_2 , we obtain a dominating set D having size at most

$$\frac{1}{2}(|D_1| + |D_2|) \leq \frac{5\alpha + 2}{10}n + \frac{29 - 20\alpha}{60}a_1 + \frac{4 - \alpha}{6}a_2.$$

From D , by inequality (3.2), we construct a desired spanning star forest that has at least

$$(3.5) \quad \begin{aligned} & n - |D| \\ & \geq \left(\frac{4}{5} - \frac{1}{2}\alpha\right)n - \frac{29-20\alpha}{60}a_1 - \frac{4-\alpha}{6}a_2 \\ & \geq \left(\frac{4}{5} - \frac{1}{2}\alpha\right)(n - a_1 - a_2) \\ & \geq \left(\frac{4}{5} - \frac{1}{2}\alpha\right)(n - \gamma(G)) \end{aligned}$$

edges since $\frac{4}{5} - \frac{1}{2}\alpha \geq \frac{29-20\alpha}{60}$, $\frac{4}{5} - \frac{1}{2}\alpha \geq \frac{4-\alpha}{6}$, and $\gamma(G) \geq a_1 + a_2$. This shows the fact stated in the theorem.

When 3 does not divide $|B_1|$ and $|B_2|$, we can modify the above argument in the same way as we have done in the proof of Theorem 3.4. For $j = 1, 2$, if $|B_j| \equiv 2 \pmod{3}$, we add two more vertices to form a 4-vertex clique with the last two

vertices in B_j ; if $|B_j| \equiv 1 \pmod{3}$, we add three more vertices to form a 4-vertex clique with the last vertex. By this modification, the inequality (3.5) becomes

$$n - |D| \geq \left(\frac{4}{5} - \frac{1}{2}\alpha\right) (n - \gamma(G)) - \frac{6 - r_1 - r_2}{6}(4\alpha - 1),$$

where $r_j \equiv |B_j| \pmod{3}$, $j = 1, 2$. Since $n - \gamma(G) \geq n/2$, the fact stated in the theorem remains true. \square

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 $NP \subset DTIME(n^{\log \log n})$ [17, 12]. Hence, intuitively, the spanning star forest problem should not be approximated within a large constant ratio in polynomial time. Now, we prove this fact rigorously.

THEOREM 3.6. *The spanning star forest problem cannot be approximated within a ratio $\frac{259}{260} + \epsilon$ in polynomial time for any $\epsilon > 0$ unless $P = NP$.*

Proof. We prove this fact using a 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 known that this problem cannot be approximated within a ratio $\frac{53}{52} - \epsilon$ for 4 regular graphs unless $P = NP$ [9]. Given a 4-regular graph $G = (V_G, E_G)$, we construct a graph $H = (V_H, E_H)$ from G by adding a length-2 path parallel to each edge in G . Without loss of generality, we assume that G is connected and not a tree. 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:

- (\star) 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 $V \subset V_H$, if it is a dominating set of H , then $(V \cap V_G) \cup \{x \in V_G \mid v_e \in V \text{ and } e = (x, y) \in E \text{ and } x < y\} \subseteq V_G$ is a vertex cover of G .

Let $\text{OPT}_{vc}(G)$, $\text{OPT}_d(H)$, and $\text{OPT}_{sf}(H)$ 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 $\text{OPT}_d(H) = \text{OPT}_{vc}(G)$. Since G is 4-regular and connected,

$$|E_G| \leq 4\text{OPT}_{vc}(G).$$

By the handshaking theorem,

$$|V_G| = \frac{1}{2}|E_G| \leq 2\text{OPT}_{vc}(G).$$

Therefore,

$$\begin{aligned} &\text{OPT}_{sf}(H) \\ &= |V_H| - \text{OPT}_d(H) = (|V_G| + |E_G|) - \text{OPT}_d(H) \\ &= (|V_G| + |E_G|) - \text{OPT}_{vc}(G) \\ &\leq 5\text{OPT}_{vc}(G). \end{aligned}$$

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 form a dominating set of H and

hence induce a vertex cover V_S of size at most k of G by the fact (\star) . Hence, since $\text{OPT}_d(H) = \text{OPT}_{vc}(G)$,

$$|V_S| - \text{OPT}_{vc}(G) \leq k - \text{OPT}_d(H) = k - (|V_H| - \text{OPT}_{sf}(H)) = \text{OPT}_{sf}(H) - |S|.$$

Thus, if $|S|$ is a ratio- $(1 - \frac{1}{5} \cdot \frac{1}{52} + \epsilon)$ approximation of the spanning star forest problem for H , where $\epsilon > 0$, i.e., $|S| \geq (1 - \frac{1}{5} \cdot \frac{1}{52} + \epsilon)\text{OPT}_{sf}(H)$, then the resulting vertex cover set V_S satisfies

$$|V_S| - \text{OPT}_{vc}(G) \leq \left(\frac{1}{5} \cdot \frac{1}{53} - \epsilon\right) \text{OPT}_{sf}(H) \leq \left(\frac{1}{52} - 5\epsilon\right) \text{OPT}_{vc}(G).$$

In other words, V_S is a ratio- $(\frac{53}{52} - 5\epsilon)$ approximation of the VERTEX COVER problem for G . This implies that the spanning star forest problem cannot be approximated within a ratio $1 - \frac{1}{5} \cdot \frac{1}{52} + \epsilon = \frac{259}{260} + \epsilon$ in polynomial time unless $P = NP$. \square

4. Algorithms for weighted graphs.

4.1. A remark on the maximum spanning star forests. In a connected weighted graph, every maximum spanning star forest may contain some isolated vertices. For example, the graph shown in Figure 4.1 has a unique spanning star graph with the maximum weight 7. Therefore, when we design an algorithm for the spanning star forest problem for connected weighted graphs, we have to consider the start 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,*

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

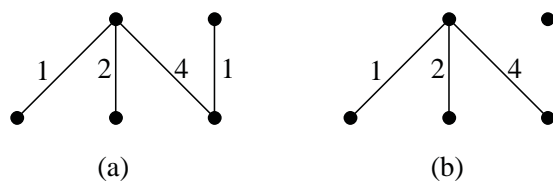


FIG. 4.1. (a) A weighted graph; (b) The unique maximum spanning star forest with an isolated vertex.

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 isolated, 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, $SFL|_v$ is a star forest (of $T(v)$) in which v is isolated or the center of a star. Hence, $SFL|_v$ 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. \square

THEOREM 4.2. *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 numbers 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 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. \square

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 this purpose. Since Step 2 also takes polynomial time, the above method can be executed in polynomial time.

Let $\text{OPT}_{sf}(G)$ and $\text{OPT}_t(G)$ 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 stars. Hence, $\text{OPT}_{sf}(G) \leq \text{OPT}_t(G)$.

Consider a rooted weighted tree T . We call a node an odd node if the unique path from the root to it has an odd number of edges and an even node otherwise. Deleting from T all edges from an odd node to an even node gives a spanning star forest of T ; deleting from T all edges from an even node to an odd node gives another

spanning star forest of T . In addition, these two resulting spanning star forests are edge-disjoint and the weight of one of these two spanning star forests is at least half of the weight of T . Hence,

$$w(SF_G) \geq \frac{1}{2}w(T_G) = \frac{1}{2}\text{OPT}_t(G) \geq \frac{1}{2}\text{OPT}_{sf}(G).$$

Therefore, the above algorithm has approximation ratio $\frac{1}{2}$. This has proved the following theorem.

THEOREM 4.3. *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 from different species, the TBA program [7] 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 alignments of multiple 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 and outputs a set of blocks (called a blockset) at the root of T . The output blockset can be considered as a packing of the pairwise alignments between sequence segments generated at each internal node.

Tandem duplication is an evolutionary event in which a genomic segment is duplicated into several adjacent copies. It is believed to be a major mechanism for producing large gene clusters. In the human and mouse genomes, a gene family may have dozens or even hundreds of members due to tandem duplication. Assume we align a set of genomes. Consider a gene family that has multiple homologous genes 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.

Currently, there are no good solutions to aligning duplication-rich genomic regions. The existing TBA program screens out duplicated alignments and thus is not able to capture all pairwise orthologous relationships in a duplicated-gene cluster. An example of this deficiency, involving the human and rat alpha-globin genes, is described in the introduction.

One way to avoid exponential growth of the number of blocks is to generate 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 $|B_v|$ to denote 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 simplicity, we considered the rows in L_P and R_P as trivial blocks. 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 may contain many different orthologous sequences, the resulting bipartite graph G_v has more than one connected component 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 for 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 , and 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.*

Nonredundancy 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 obvious. We prove the nonredundancy 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; 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 incident only to e . Then, the segment corresponding to x is covered only 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'' is not a leaf, by induction, we assume that both $B_{v'}$ and $B_{v''}$ satisfy the nonredundancy 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 appears only 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. \square

Theorem 5.1 can be used to estimate the base-pair size $\|B\|$ of the blockset B generated by the algorithm. Let N be the number of sequences in the input genomic sequence set S . Since any pair of orthologous sequence segments s and s' can be assumed to have more than 50% identical bases, they have roughly equal length; i.e., $|s| = \Theta(|s'|)$. For each block $X \in B$, we use $r_{\text{unique}}(X)$ to denote the unique row in X . By assumption, the base-pair size $|X|$ of X is at most $N\Theta(|r_{\text{unique}}(X)|)$ since X has at most N rows. Therefore, using the fact that $r_{\text{unique}}(X)$'s are disjoint segments of the input sequences in S , we have

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

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

An initial test shows that our proposed approach is quite promising. For example, we consider the alpha-globin gene cluster that has as many as 91 genes in 20 mammals. Both alpha-related and theta-related genes have clear many-to-many homologous relationships. When aligning the alpha-globin gene cluster region (of total length 3.9 Mbytes) in 20 mammals, a straightforward strategy meeting the two requirements described in the last theorem produced a blockset of size over 900 Mbytes, but the preliminary implementation of our approach can reduce the alignment size to less than 10 Mbytes with little sacrifice of the alignment quality.

6. Conclusions. In this paper, we initiate the algorithmic study of the spanning star forest problem. In particular, we present a $\frac{3}{5}$ -approximation algorithm for 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 $\frac{259}{260} + \epsilon$. This study raises several research questions for future investigation. For example, how to obtain an approximation algorithm with better ratio and how to improve the inapproximability result are two interesting open problems. After this work was submitted, a polynomial-time algorithm with approximation ratio 0.71 was presented in [8]. Another interesting question is how to design approximation algorithms for the spanning star forest problem for directed and weighted graphs.

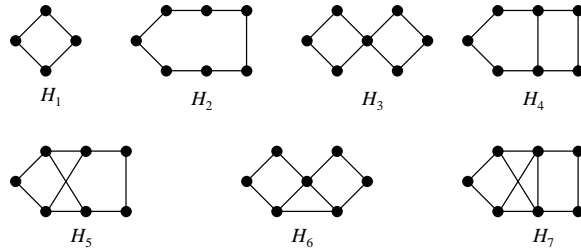


FIG. A.1. Seven bad graphs whose minimum dominating sets have size larger than $2n/5$ (reproduced from [18]; reprinted with permission of John Wiley & Sons, Inc.).

Appendix A. Finding a dominating set of size at most $2n/5$.

THEOREM A.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 A.1 (called bad graphs), then $\gamma(G) \leq \frac{2}{5}|V|$. 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 $G = (V, E)$. We prove by induction on $|V|$. It is easy to check that, for $|V| \leq 15$, we can always find a desired dominating set. Now assume that $|V| > 15$ and that the theorem holds for all graphs G' that have a smaller number of vertices than G and that are not listed in Figure A.1. We show how to construct a dominating set of G whose size is at most $\frac{2}{5}|V|$.

If G is not edge-minimal with respect to connectedness and with a 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 $B(G) = \{u \in V \mid d(u) > 2\}$. 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 such a dominating set contains at most $\frac{2}{5}|V|$ vertices for any cycles that are different from H_1 and H_2 .

If $|B(G)| = 1$, then G is a union of cycles that intersect at the unique vertex $u \in B(G)$. Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be the sets of cycles of length 4, 7, and another length, 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 that contains u and is of size at most $\frac{2}{5}|C_i|$ for each i ; (iv) merge all the dominating sets. The size of this dominating set is

$$1 + |\mathcal{A}| + 2|\mathcal{B}| + \sum_{i=1}^t \left(\frac{2}{5}n_i - 1 \right) = 1 + |\mathcal{A}| + 2|\mathcal{B}| + \frac{2}{5} \sum_{i=1}^t n_i - t,$$

in which n_i denotes the number of vertices in C_i . Noting that $|V| = 3|\mathcal{A}| + 6|\mathcal{B}| + \sum_{i=1}^t n_i - t + 1$, we have that the size of the constructed dominating set is larger than $\frac{2}{5}|V|$ only if $G = H_3$, which contains exactly two length-4 cycles.

Now assume that $|B(G)| \geq 2$. In the following, we call G a good graph if it is not listed in Figure A.1. We consider the following cases.

Case 1. *There is an edge uv between two vertices u and v 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

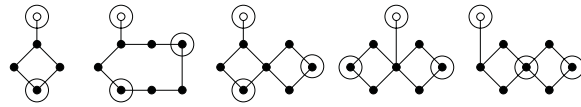


FIG. A.2. A graph formed by adding a vertex to a bad graph contains one of these graphs. The white vertex is the additional vertex. The circles mark the vertices in a dominating set of each graph.

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. By hypothesis, we can find dominating sets D_u and D_v of G_u and G_v such that $|D_u| \leq \frac{2}{5}|V_u|$ and $|D_v| \leq \frac{2}{5}|V_v|$. $D_u \cup D_v$ contains at most $\frac{2}{5}|V|$ vertices and dominates G .

Case 1.2. Only one, say, G_v is a good graph. Then G_u is a bad graph. Let $G_u = (V_u, E_u)$, and let G_u^* be the graph with vertex set $V_u^* = V_u \cup \{v\}$ and edge set $E_u^* = E_u \cup \{uv\}$. We also let 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 $\delta(G_v^*) \geq 2$. Since G_u is a bad graph, G_u^* contains one of the graphs in Figure A.2. Note that H_4, H_5, H_7 in Figure A.1 contain H_2 and H_6 contains H_3 , and so we need only to consider the cases in Figure A.2. From this figure, we can see that G_u^* has a dominating set D_u^* that contains v and satisfies that $|D_u^*| \leq \frac{2}{5}|V_u^*|$. Since G_u is bad and $|V| \geq 16$, G_v^* contains at least eight vertices, and so it is good. By induction, it has a dominating set D_v^* of size at most $\frac{2}{5}|V_v^*|$. Then $D_u^* \cup D_v^*$ form a dominating set of G whose size is at most $\frac{2}{5}|V|$.

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 that contain exactly one vertex in $B(G)$, and this unique vertex is called the *endpoint* of the cycle. We will also use 2-paths to refer to the paths whose endpoints are in $B(G)$ and in which other vertices are of degree 2.

If there is a path $vv_1v_2v_3v'$ in G such that $v \neq v'$ and $d(v_1) = d(v_2) = d(v_3) = 2$, we construct a graph G' by removing v_1, v_2, v_3 and adding the edge vv' if $vv' \notin E(G)$. Clearly G' is connected and $\delta(G') \geq 2$. Since $|V| > 16$, G' must be good. Hence, G' has a dominating set D' of size at most $\frac{2}{5}|V'|$. We construct a dominating set D for G as follows: (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 $|D| = |D'| + 1 \leq \frac{2}{5}(|V'| + 3) = \frac{2}{5}|V|$. Now assume that all 2-cycles in G are of length at most 4 and all 2-paths in G are of length at most 3.

Case 2.1. G contains a 2-cycle of length 4 whose endpoint's degree is larger than 3. Let this cycle be $vv_1v_2v_3v$, where v is the endpoint. A graph G' is formed by deleting v_1, v_2 , and v_3 from G . Since G' contains at least thirteen vertices, it is good and has a dominating set D' of size at most $\frac{2}{5}|V'|$. Then $D' \cup \{v_2\}$ is a dominating set of G whose size is smaller than $\frac{2}{5}(|V'| + 3) = \frac{2}{5}|V|$.

Case 2.2. Every 2-cycle of length 4 in G has an endpoint of degree 3. Consider such a cycle $vv_1v_2v_3v$, where v is the endpoint. 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.

In this case, we assume that G contains a 4-cluster, say, $C = \{v, v_1, v_2, v_3, v'\}$, with anchor v' such that the neighbor u of v' not in C has degree larger than 2 in G , and we define $G' = G - C$. Then G' is a connected graph with $\delta(G) \geq 2$. Since G' contains at least eleven vertices, G' is good, and it has a dominating set D' such that

$|D'| \leq \frac{2}{5}|V'|$. Furthermore, $D = D' \cup \{v, v_2\}$ is a dominating set of G and $|D| \leq \frac{2}{5}|V|$.

Case 2.3. For every 4-cluster, its anchor's neighbor that is not in the cluster is of degree 2. We define a 2-cycle of length 3 in G to be a 3-cluster, and the anchor of a 3-cluster is the endpoint of the cluster. Note that we define the anchor of a 3-cluster differently. Finally, each vertex u such that $d(u) > 2$ and u is not an endpoint of any 2-cycles is considered as a 1-cluster with u as the anchor.

If there is an edge joining two anchors, the two anchors must be that of two 4-clusters since other cases have been considered in Case 2.2. In this case, this graph has only ten vertices. But we assume G has at least sixteen vertices. Hence, there is no edge joining two anchors in G . A dominating set D of G is formed by collecting all anchors and one vertex in each 4-cluster. Now we prove that $|D| \leq \frac{2}{5}|V|$.

Let m_4, m_3, m_1 be the number of 4-clusters, 3-clusters, and 1-clusters, respectively. Then, there are also $m_4, m_3,$ and m_1 anchors of 4-clusters, 3-clusters, and 1-clusters, respectively. Clearly, $|D| = 2m_4 + m_3 + m_1$.

Now we count the nonanchor vertices. Each 4-cluster, 3-cluster, and 1-cluster contains 4, 2, and 0 nonanchor vertices, respectively. Furthermore, on each 2-path connecting two anchors, there are one or two nonanchor vertices depending on whether it is of length two or three. Therefore, the number of nonanchor vertices on these paths is at least $\frac{1}{2}(m_4 + m_3 + 3m_1)$ since there is at least one nonanchor vertex in the 2-path between two anchors and the degree of the unique vertex in a 1-cluster is at least 3. Hence, the number of nonanchor vertices is at least $\frac{9}{2}m_4 + \frac{5}{2}m_3 + \frac{3}{2}m_1$. Recall that the dominating set D contains one nonanchor vertex in each 4-cluster. This implies $|V| \geq \frac{11}{2}m_4 + \frac{7}{2}m_3 + \frac{5}{2}m_1$. Simple computation shows that $|D| \leq \frac{2}{5}|V|$. \square

A recursive algorithm to compute a desired dominating set follows from the above induction proof. To implement this algorithm, we have to compute $d(v)$ for each vertex in G and find (a) an edge between two vertices in $B(G)$, (b) a length-4 path that contains three consecutive degree-2 vertices in the middle, and (c) a 4-cluster with an endpoint of degree larger than 3. Given the adjacent matrix of a graph G as input, we can find the degree of a vertex in $O(|V|)$ time and thus find $B(G)$ in $O(|V|^2)$ time. With $B(G)$, we can find an edge between two vertices in $B(G)$ if any exist in $O(|V|^2)$ time. To compute (b), we just need to check if there is a degree-2 vertex having two degree-2 neighbors and these two neighbors have different neighbors in $O(|E|^2)$ time. The task (c) can be done similarly. Overall, the algorithm finds a dominating set iteratively. The number of vertices decreases by at least 1 after each iteration step, and each iteration step takes $O(|E|^2)$ time and removes three vertices from the graph. Therefore, the running time of the algorithm is $O(|E|^2|V|)$.

Acknowledgment. The authors would like to thank the anonymous reviewers for various useful suggestions for improving the presentation of this work.

REFERENCES

- [1] A. AGRA, D. CARDOSO, O. CERDEIRA, AND E. ROCHA, *A Spanning Star Forest Model for the Diversity Problem in Automobile Industry*, manuscript, 2005.
- [2] J. AKIYAMA AND M. KANO, *Path factors of a graph*, in Graph Theory and Its Applications, F. Haray and J. Maybee, eds., Wiley, New York, 1984, pp. 1–21.
- [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. ACM, 41 (1994), pp. 153–180.

- [6] V. BERRY, S. GUILLEMOT, F. NICOLAS, AND C. PAUL, *On the approximation of computing evolutionary trees*, in Proceedings of the 11th Annual International Computing and Combinatorics Conference, Lecture Notes in Comput. Sci. 3595, Springer-Verlag, New York, 2005, pp. 115–125.
- [7] W. J. KENT, C. RIEMER, L. ELNITSKI, A. F. SMIT, K. M. ROSKIN, R. BAERTSCH, K. ROSENBLUM, H. CLAWSON, E. D. GREEN, D. HAUSSLER, AND W. MILLER, *Aligning multiple genomic sequences with the threaded blockset aligner*, Genome Res., 14 (2004), pp. 708–715.
- [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 Proceedings of APPROX 2007, Lecture Notes in Comput. Sci. 4627, Springer-Verlag, New York, pp. 44–58.
- [9] M. CHLEBÍK AND J. CHLEBÍKOVÁ, *Inapproximability results for bounded variants of optimization problems*, Theoret. Comput. Sci., 354 (2006), pp. 320–338.
- [10] E. J. COCKAYNE, S. E. GOODMAN, AND S. T. HEDETNIEMI, *A linear algorithm for the domination number of a tree*, Inform. Process. Lett., 4 (1975), pp. 41–44.
- [11] 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.
- [12] U. FEIGE, *A threshold of $\ln n$ for approximating set cover*, J. ACM, 45 (1998), pp. 634–652.
- [13] S. FERNEYHOUGH, R. HAAS, D. HANNSON, AND G. MACGILLIVRAY, *Star forests, dominating sets and Ramsey-type problems*, Discrete Math., 245 (2002), pp. 255–262.
- [14] S. I. HAKIMI, J. MITCHEM, AND E. SCHMEICHEL, *Star arboricity of graphs*, Discrete Math., 149 (1996), pp. 93–98.
- [15] T. HAYNES, S. T. HEDETNIEMI, AND P. J. SLATER, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.
- [16] M. M. HOU, P. BERMAN, L. X. ZHANG, AND W. MILLER, *Controlling size when aligning multiple genomic sequences with duplications*, in Proceedings of WABI (Zurich, 2006), Springer-Verlag, Berlin, 2006, pp. 138–149.
- [17] C. LUND AND M. YANNAKAKIS, *On the hardness of approximation minimization problems*, J. ACM, 41 (1994), pp. 961–981.
- [18] W. MCCUAIG AND B. SHEPHERD, *Domination in graphs with minimum degree two*, J. Graph Theory, 13 (1989), pp. 749–762.
- [19] B. REED, *Paths, stars and the number three*, Combin. Probab. Comput., 5 (1996), pp. 277–295.