The maximum weight spanning star forest problem on cactus graphs

Viet Hung Nguyen

The maximum weight spanning star forest problem on cactus graphs

Viet Hung Nguyen
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6
4 place Jussieu, Paris, France
Hung.Nguyen@lip6.fr

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year
Published Day Month Year

A star is a graph in which some node is incident with every edge of the graph, i.e. a graph of diameter at most 2. A star forest is a graph in which each connected component is a star. Given a connected graph G in which the edges may be weighted positively. A spanning star forest of G is a subgraph of G which is a star forest spanning the nodes of G. The size of a spanning star forest F of G is defined to be the number of edges of F if G is unweighted and the total weight of all edges of F if G is weighted. We are interested in the problem of finding a Maximum Weight spanning Star Forest (MWSFP) in G. In [7], the authors introduced the MWSFP and proved its NP-hardness. They also gave a polynomial time algorithm for the MWSF problem when G is a tree.
In this paper, we present a linear time algorithm that solves the MSWF problem when G is a cactus.

Keywords: star forest; dominating set; cactus graphs; genomic sequence alignment

Mathematics Subject Classification: 05C85, 90C27, 90C39

1. Introduction

A *star* is a graph in which some node is incident with every edge of the graph, i.e. a graph of diameter at most 2. In particular, an isolated node is also a star. A star in G is a *multiple-node star* if it is not an isolated node. For a given multiple-node star, we designate the *center* of the star as the node of degree strictly grater than 1 or any of the nodes if the star is an edge. A node in a multiple-node star which is not the center is a *leaf*. A *star forest* is a graph in which each connected component is a star. Given a connected graph G in which the edges may be weighted positively. A *spanning star forest* of G is a subgraph of G which is a star forest spanning the nodes of G. Note that a spanning star forest can contain isolated nodes. The size of a spanning star forest F of G is defined to be the number of edges of F if G is unweighted and the total weight of all edges of F if G is weighted. We are interested in the problem of finding a maximum weight spanning star forest in G. As we can

take isolated nodes, any maximum weight star forest can be extended without additional weight to a spanning star forest. Hence, without loss of generality, we shall focus on the problem of finding a Maximum Weight Star Forest ($MWSFP$) in G. The MWSFP is NP-hard already for the case G is unweighted. In fact, in this case, since G is connected, there always exists a maximum size spanning star forest F of G that does not contain isolated nodes. A *dominating set* of a graph is a subset of the nodes such that every other node is adjacent to a node in the dominating set. Observe that in a maximum star forest which does not contain isolated nodes, each node is either a center or adjacent to a center. Hence the set of centers form a dominating set of the graph. Therefore, the size of the maximum star forest is the number of nodes minus the size of the minimum dominating set. Computing the maximum star forest of a graph is NP-hard because computing the minimum dominating set is NP-hard. The spanning star forest problem has found applications in computational biology. Nguyen et al. [7] use the spanning star forest problem to give an algorithm for the problem of aligning multiple genomic sequences, which is a basic bioinformatics task in comparative genomics. The spanning star forest problem and its directed version have found applications in the comparison of phylogenetic trees [5] and the diversity problem in the automobile industry [1].

The unweighted version of MWSFP has been investigated intensively in recent years. Nguyen et al. [7] prove that the problem is APX-hard by presenting an explicit inapproximability bound of 259/260 and present a combinatorial 0.6-approximation algorithm. Optimal polynomial-time algorithms are presented for special graph classes such as planar graphs and trees. Chen et al. [4] present a better algorithm with approximation ratio 0.71. Later, Athanassopoulos et al. [2] improve this approximation ratio to 0.803 by using the fact that the problem is a special case of complementary set cover . Interesting generalizations include node-weighted and edge-weighted versions of SSF. [7] and [4] present approximation algorithms and APX-hardness results for these problems as well. Stronger inapproximability results for these problems recently appeared in [6] and [3].

Contrary to the unweighted version, the only work on the general weighted version of MWSFP (that we will call shortly the MWSFP from now on) is the paper of Nguyen et al. [7]. The authors prove that one can solve the MWSFP in linear time when G is a tree. Based on this results, they design a simple $\frac{1}{2}$-approximation algorithm for MWSFP for the general case.

In this paper, we present a linear time algorithm that solves the MSWF problem when G is a cactus. The algorithm is a combination of the idea about the values to be computed at each node of the algorithm for trees in [7] and our idea to establish an order over the nodes in which one can simulate the construction of any star forest and to use longest paths computation in acyclic directed graphs for handling the cycles.

2. Linear time algorithm for MWSFP when G is a cactus

In this section, $G = (V, E)$ is a cactus in which the edges are weighted by a vector $c \in \mathbb{R}_+^E$. A *cut node* in G is a node whose deletion makes G disconnect. A *pendant node* is a node degree 1 in G. A *block* of a graph is a maximal connected subgraph of G that has no cut-node. As G is cactus, every block in G is a simple cycle or a single edge.

2.1. *Building a tree of cut nodes, pendant nodes and cycles*

The first step of the algorithm is to build a tree τ whose (pseudo)-nodes (we call them pseudo-nodes in order to distinguish with the nodes in G) correspond to the cut nodes, the pendant nodes and the cycles in G. We will use the algorithm specified in [8] for detecting the blocks of G to build τ. The algorithm which runs in $O(|E|)$, builds a depth-first search tree T of G and discards portions of T as blocks are identified. Each time a block B of G together with its highest ancestor node w with respect to T are detected by the algorithm, we create the correspondences of B and w in τ as follows.

- if B is an edge vw then let us create two singleton pseudo-nodes v and w in τ with w is the father of v in τ,
- otherwise, i.e. B is a cycle. Then we create one cycle pseudo-node B and one singleton pseudo-node w in τ where w is the father of B. Let us set w as *the root node* of B. Moreover, for each cut node v in B, let us set B as the father of v in τ (as T is a depth-first tree, the pseudo node v has been already in τ but its father have not been determined yet).

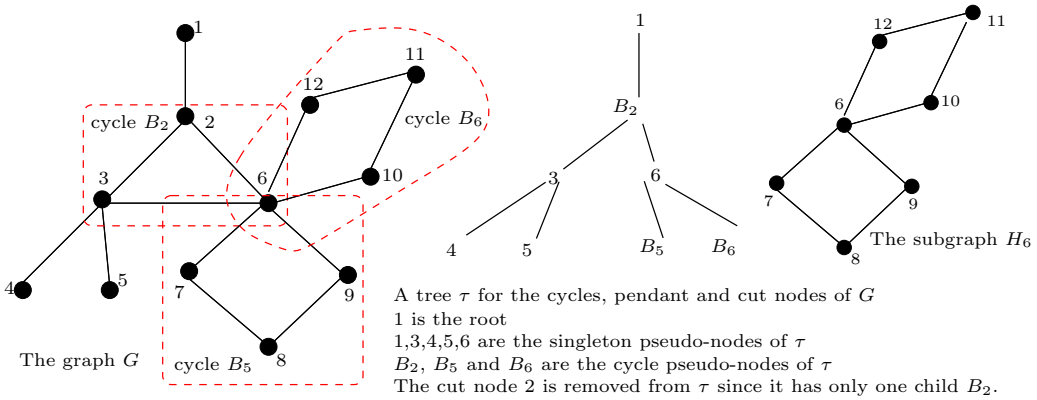

Fig. 1. A cactus graph G, one of its tree τ and a subgraph induced from a pseudo-node in τ and its descendants.

At the end of the above construction, for each singleton pseudo-node w in τ which has only one son and the latter is a cycle pseudo-node B, remove w and link the node B to the the father of w in τ (see Figure 1 for an example of τ). It is easy to see that these additional works do not change the linear time complexity of the algorithm in [8].
Let N be a pseudo node in τ, let $u_N \in V$ be

- the root node of N if N is a cycle pseudo-node in τ,
- the node N itself, otherwise.

Let H_N be the subgraph of G induced by the nodes in G corresponding to the subtree rooted in N of τ (see Figure 1 for an example). For each pseudo-node N, we compute three values:

- $\Phi(N)$: The maximum weight of a spanning star forest of H_N in which u_N is a center of a multiple-node star.
- $\Psi(N)$: The maximum weight of a spanning star forest of H_N in which u_N is a leaf of a multiple-node star.
- $\Omega(N)$: The maximum weight of a spanning star forest of H_N in which u_N is an isolated node.

We call these values, *the elementary values* associated to N.

2.2. *N is a singleton pseudo-node in τ*

Suppose that N corresponds to a node w in G. Let $CS(N)$ be the subset of the children of N in τ which are singleton pseudo-node. Let $CC(N)$ be the subset of the children of n in τ which are cycle pseudo-node. For each singleton pseudo-node $v \in CS(N)$, we define a cost $c(N, v)$ which is equal to the cost c_{wv} of the edge $vw \in E$. We now specify how to compute the values of $\Phi(N)$, $\Psi(N)$ and $\Omega(N)$ provided that those of the descendants of N have been computed. Before this, let us note that there may be some confusion between $\Phi(N)$ and $\Psi(N)$. Indeed, when the star forest contains a star which is exactly some edge wv with $v \in H_N$ then one may consider this as a multiple-node star centered either at w or at v. Thus this case is considered two times, one in the computation of $\Phi(N)$ and the other in the computation of $\Psi(N)$. To avoid this, we consider that this type of star is always centered at the father with respect to τ, hence at w as it happens. We have the following proposition which is partly inspired from [7].

Proposition 2.1. *If N is a singleton pseudo-node, we have*

$$\Omega(N) = \sum_{B\in CC(N)} \Omega(B) + \sum_{v\in CS(N)} \max\{\Phi(v), \Psi(v), \Omega(v)\},$$

$$\Psi(N) = \max[\max_{B\in CC(N)} (\Psi(B) + \sum_{B'\in \mathcal{CC}(N)\setminus\{B\}} \Omega(B') + \sum_{v\in CS(N)} \max\{\Phi(x), \Psi(x), \Omega(x)\}),$$
$$\max_{v\in CS(N)} (c(N,v) + \Phi(v) + \sum_{B\in CC(N)} \Omega(B) + \sum_{x\in CS(N)\setminus\{v\}} \max\{\Phi(x), \Psi(x), \Omega(x)\})],$$

$$\Phi(N) = \max[\max_{B\in CC(N)} (\Phi(B) + \sum_{B'\in CC(N)\setminus B} \max(\Phi(B'), \Omega(B')) + \sum_{v\in CS(N)} \Delta(v)),$$
$$\sum_{B\in CC(N)} \Omega(B) + \sum_{v\in CS(N)} \Delta(v) - \min_{v\in CS(N)} (\Delta(v) - \Omega(v) - c(N,v))],$$

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

Proof. We can see easily the validity of the formulas for $\Omega(N)$. A maximum weight star forest in H_N where N is an isolated node is composed by maximum weight star forests in H_B where u_B (note that $u_B = N$) is an isolated node for all $B \in CC(N)$ plus maximum weight star forests in H_v for all $v \in CS(N)$ (note that $v \neq N$).
The formula for $\Psi(N)$ takes into account two possible cases:

- N is a leaf of a star centered at a node in H_B for some cycle B with N as root node,
- N is a leaf of a star centered at a cut node v. In this case, one must include the edge Nv in the star forest.

We can see that there are two possible cases for a star centered at N:

- there exists a cycle B in $CC(N)$ such that there is a (sub)star centered at $u_B = N$ in H_B,
- otherwise, i.e. there exists a $v \in CS(N)$ such that the star contains the edge Nv.

The two cases are considered respectively in first and second parts of the formula for $\Phi(N)$. Note that for the second case, one can remark that the star forest in H_N corresponding to the value $\Phi(N)$ contains necessarily the edge $u_N v$ and the subset of edges associated to $\Omega(v)$ for $v = \text{argmin}_{v\in\mathcal{C}(N)}(\Delta(v) - \Omega(v) - c(N,v))$.
By a careful implementation of these formulas, we can compute these values in $O(|CC(N)| + |CS(N)|)$. □

When computing each of the three above values, we also keep in memory a subset of edges associated with them which is the union of the subset of edges associated with the elementary values (of the children) and possibly the edge Nv for some $v \in \mathcal{C}(N)$ if these elementary values and $c(N,v)$ involves in the exact computation of the value in question.

2.3. *N is a cycle pseudo-node in τ*

Let us consider the case when N is a cycle pseudo-node. Let C denote the cycle in G corresponding to N and the nodes in C are numbered from 1 to n with $n = u_N$, the root node of N. For each node $i \neq n$ in C, if i is a cut node, i.e. i is in τ, we have computed the values $\Phi(i)$, $\Psi(i)$ and $\Omega(i)$. Otherwise, let us set $\Phi(i) = \Psi(i) = \Omega(i) = 0$. Let e_i with c_i as weight denote the edge between the node i and the node $i+1$ for $i = 1, \ldots, n-1$. In particular, the edge e_n with c_n as weight denotes the edge between n and 1. For a node $i \in C \setminus \{n\}$, let $H_{1,i}$ be the subgraph induced by the nodes 1, ...,i and their descendants. For the node n, let $H_{1,n}$ be the subgraph induced by the nodes 1, ...,n and their descendants except the edge e_n. We build a graph G'_C from C as follows. For every $i \in C$, we create in G'_C three nodes i_0, i_1 and i_2 called the clones of i. The indexes in these nodes simulates the possible choices for any star forest S provided that the part of this star forest in the subgraph of G induced by the descendants of B has been determined. We check the edges of C in the order e_1, e_2, ..., e_{n-1} (the edge e_n will be considered later) and examine all the possibilities of include or do not include each of them in S. We can see that by this order of choices, the choice of including or not e_{i-1} in S depend on the choices already done for the subgraph $H_{1,i-1}$ and on the fact that i is an isolated node, a center or a leaf in $S \cap H_{1,i}$. Hence, the clone i_0 represents the cases in which i is an isolated node in $S \cap H_{1,i}$. The clone i_1 represents the case when i is a center of a multiple-node star in $S \cap H_{1,i}$. At last, the clone i_2 represents the case when i is a leaf of a multiple-node star in $S \cap H_{1,i}$. Note that when i belong to a star in $S \cap H_{1,i}$ which is an edge ik where k is a descendant of i or the node $i-1$, we consider that i is always the center and k is the leaf. Hence, more precisely, the clone i_2 represents the case when i is a leaf of a multiple-node star which has at least two leaves in $S \cap H_{1,i}$. The arcs in G'_C are created as follows: for the clones of every node $2 \leq i \leq n$ in G,

- we create three arcs $((i-1)_0, (i)_0)$, $((i-1)_1, (i)_0)$, $((i-1)_2, (i)_0)$ with cost $\Omega(i)$. This represents the cases when i is an isolated node in $S \cap H_{1,i}$.
- we create an arc $((i-1)_0, i_1)$ with cost $c_{i-1} + \max(\Phi(i), \Omega(i))$. This represents the cases when $i-1$ is an isolated in $S \cap H_{1,i-1}$ and i is a center of a star containing the edge e_{i-1} in $S \cap H_{1,i}$. We create also two arcs $((i-1)_1, i_1)$ and $((i-1)_2, (i)_1)$ with cost $\Phi(i)$. This represents the cases when i is a center of a star not containing the edge e_{i-1} in $S \cap H_{1,i}$. Since we look for a maximum weight star forest, in these cases, $(i-1)$ can not be an isolated node in $S \cap H_{1,i-1}$ (since e_{i-1} should be in S otherwise).
- an arc $((i-1)_1, i_2)$ with cost $c_{i-1} + \Omega(i)$. This represents the cases when i is a leaf of a star in $S \cap H_{1,i}$ whose center is $i-1$. We create also two arcs $((i-1)_0, i_2)$ and $((i-1)_1, (i)_2)$ $((i-1)_2, (i)_2)$ with cost $\Psi(i)$. This represents the cases when i is a leaf of a star in $S \cap H_{1,i}$ which does not contain the edge e_{i-1}.

Remark 2.2. The graph G'_C is acyclic and the number of nodes and edges in G'_C is $O(|C|)$.

In G'_C, we compute the costs C_{pq} of the nine longest paths (in term of cost c) from 1_p to n_q respectively for $p, q = 0, 1, 2$. By Remark 2.2, these computations can be done in $O(|C|)$.
Let H be the subgraph of G induced by the node 1 without its descendants and all the nodes $2, 3, \ldots, n$ and their descendants, except the edge e_n. Let H_1 be the subgraph induced by the descendants of 1.

Lemma 2.3. *Given any maximum star forest S of G, C_{pq} is the maximum cost that S can take in H under the hypothesis that*

H1 *If $p = 0$, the node 1 is an isolated node in $H_1 \cap S$. If $p = 1$ the node 1 is an center of a star in $H_1 \cap S$. If $p = 2$, the node 1 is a leaf of a star in $H_1 \cap S$.*
H2 *If $q = 0$ then the node n is an isolated node in $H \cap S$. If $q = 1$ the node n is an center of a star in $H \cap S$. If $q = 2$, the node n is a leaf of a star in $H \cap S$.*

Proof. The graph G'_C is built to simulate the construction of a star forest S in H under the hypothesis that the values $\Omega(i)$, $\Phi(i)$ and $\Psi(i)$ are known for all $1 \leq i \leq n$. We decide which edges in C will belong to S. Precisely, for the moment, we want to decide which edges in C except e_n will belong to S. The arcs from $(i-1)_p$ to i_q simulate all the possibilities of taking or not the edge e_i into S in the order of $e_1, e_2, \ldots, e_{n-1}$. Thus there is an one-one correspondence between any star forest S in H and a path from 1_p to n_q in G'_C. For example, if the cost of an arc chosen is one of the values $\Omega(i+1)$, $\Phi(i+1)$ and $\Psi(i+1)$ then the arc e_i will not be included in S and on the contrary, the star forest in the subgraph induced by the descendants of $(i+1)$ having the same value is included in S. If the cost of the arc is c_i + one of the values $\Omega(i+1)$ and $\Phi(i+1)$ then the arc e_i will be included in S and the star forest in the subgraph induced by the descendants of $(i+1)$ having the same value is also included in S. It is thus obvious that the longest path correspond to the maximum star forest. □

It remains now to include the choices for e_n and the elementary values $\Phi(1)$, $\Psi(1)$ and $\Omega(1)$ in the computation for $\Omega(B)$, $\Phi(B)$ and $\Psi(B)$, the elementary values associated with the pseudo-node B in τ.

Proposition 2.4.

$$\Omega(B) = \max(C_{00} + \Omega(1), C_{10} + \Phi(1), C_{20} + \Psi(1)),$$

$$\Phi(B) = \max(C_{01} + \Omega(1), C_{11} + \Phi(1), C_{21} + \Psi(1), C_{00} + c_n + \Omega(1), C_{01} + c_n + \Omega(1)$$

$$\Psi(B) = \max(C_{02} + \Omega(1), C_{12} + \Phi(1), C_{22} + \Psi(1), C_{10} + c_n + \Phi(1))$$

Proof. Let H_B be the subgraph induced by B and its descendants. The value of the maximum star forest in H_B in which u_B is an isolated node is $\Omega(B)$. Such a star forest in H_B is clearly composed by a star forest in H (defined before Lemma 2.3) containing no edge incident to the node n and and an appropriate edge subset associated with one of the elementary values $\Omega(1)$, $\Phi(1)$ and $\Psi(1)$. Hence, by Lemma 2.3 the value of this star forest is represented by one of the values of a path from a node 1_p with $p = 0, 1, 2$ to the destination n_0 + an appropriate value taken from the values $\Omega(1)$, $\Phi(1)$ and $\Psi(1)$. Hence, this justifies the formula:

$$\Omega(B) = \max(C_{00} + \Omega(1), C_{10} + \Phi(1), C_{20} + \Psi(1)),$$

Note that in this formula, the edge e_n is not included in the star forest, this guarantees that u_B is an isolated node.
The value of the maximum star forest in H_B in which u_B is a center of a multiple-node star is $\Phi(B)$. Any star forest in H_B in which u_B is a center of a multiple-node star is represented by one of the following cases:

- the multiple-node star does not contain e_n. The value of such a star is the value of a path from a node 1_p with $p = 0, 1, 2$ to the destination n_1 (see Lemma 2.3)+ an appropriate value taken from the values $\Omega(1)$, $\Phi(1)$.
- the multiple-node star contains e_n. The value of such a star by the value of a path from the node 1_0 to the destination n_0 or n_1 (see Lemma 2.3) + $\Omega(1)$+ c_n.

Hence, we have $\Phi(B) = \max(C_{01}+\Omega(1), C_{11}+\Phi(1), C_{21}+\Psi(1), C_{00}+c_n+\Omega(1), C_{01}+c_n+\Omega(1))$. The value of the maximum star forest in H_B in which u_B is a leaf of multiple-node star is $\Psi(u_B)$. Any star forest of this type is clearly represented by

- either the value a path from a node 1_p with $p = 0, 1, 2$ to the destination n_2 (see Lemma 2.3) + an appropriate value taken from the values $\Omega(1)$, $\Phi(1)$, and $\Psi(1)$.
- or the value of a path from a node 1_1 to n_0 (see Lemma 2.3) + $\Phi(1)$ + c_n.

Hence, we have $\Psi(B) = \max(C_{02} + \Omega(1), C_{12} + \Phi(1), C_{22} + \Psi(1), C_{10} + c_n + \Phi(1)$□

The subset associated with each elementary value is the union of the edges of the cycle C taken by the longest path and the subsets of edges associated with the elementary values involving in the exact computation of the elementary value in question.

2.4. *Algorithm*

We are now ready to state the algorithm.

Step 1. Building the tree τ as specified in Section 2.1. Set r be the root pseudo-node of τ.

Step 2. Evaluate the elementary values $\Phi(v)$, $\Psi(v)$ and $\Omega(v)$ for all pseudo-nodes $v \in \tau$ in left-right and bottom-up order. If the current pseudo-node v correspond to a singleton in G, evaluate the elementary values as specified in Section 2.2. Otherwise, i.e. v correspond to a cycle in G, evaluate the elementary values as specified in Section 2.3. In all the cases, memorize the selected edges.

Step 3. Set $S \leftarrow \max\{\Phi(r), \Psi(r), \Omega(r)\}$ and output the subset of edges as associated with S as the maximum star forest.

Theorem 2.5. *The algorithm gives a maximum weight star forest in $O(|E|)$ time when G is a cactus.*

Proof. Propositions 2.1 and 2.4 show the validity of the elementary values $\Phi(r), \Psi(r), \Omega(r)$. Since any star forest in G can be reconstructed by a simulation (represented maximally by the elementary values) in the pseudo-nodes of τ in some left-right and bottom-up mode with respect to the root r, by taking $S \leftarrow \max\{\Phi(r), \Psi(r), \Omega(r)\}$, the algorithm output a maximum weight star forest. As we explain above, Step 1. could be performed in $O(|E|)$ time. The computation time of the elementary values associated to each pseudo-node B in τ is either $O(|CC(B)| + |CS(B)|)$ if the pseudo-node is a singleton and $O(|C|)$ if B is a cycle C. Hence, overall the time complexity of the algorithm is $O(|E|)$. □

Concluding remarks

We have presented a linear time algorithm for the maximum weight star forest for cactus graphs. This was made possible by establishing an order over the nodes in which one can simulate the construction of any star forest. One may hope to extend the algorithm in this paper for graphs which owns a certain natural order on the nodes, e.g. triangulated graphs.

References

[1] A. Agra, D. Cardoso, O. Cerfeira, and E. Rocha. A spanning star forest model for the diversity problem in automobile industry. In ECCO XVIII, Minsk, May 2005.

[2] S. Athanassopoulos, I. Caragiannis, C. Kaklamanis and M. Kyropoulou. An Improved Approximation Bound for Spanning Star Forest and Color Saving. In MFCS, pages 90-101, 2009.

[3] D. Chakrabarty and G. Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. In FOCS, pages 687-696, 2008.

[4] N. Chen, R. Engelberg, C. T. Nguyen, P. Raghavendra, A. Rudra, and G. Singh. Improved Approximation Algorithms for the Spanning Star Forest Problem. In APPROX/RANDOM, pages 44-58, 2007.

[5] V. Berry, S. Guillemot, F. Nicholas, and C. Paul. On the approximation of computing evolutionary trees. In Proceedings of the Eleventh Annual International Computing and Combinatorics Conference, pages 115-123, 2005.

[6] J.He and H. Liang. On Variants of the Spanning Star Forest Problem. In FAW-AAIM, pages 70-81, 2011.
[7] C.T. Nguyen., J. Shen, M. Hou, L. Sheng, W. Miller, and L. Zhang. Approximating the spanning star forest problem and its applications to genomic sequence alignment. *SIAM J. Comput.*, Vol. 38, No.3, pages 946-962, 2008.
[8] D. West. Introduction to graph theory. Prentice Hall, 2000.