JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 8, Number 1, 2001
Mary Ann Liebert, Inc.
Pp. 69–78

Perfect Phylogenetic Networks with Recombination

LUSHENG WANG,[1] KAIZHONG ZHANG,[2] and LOUXIN ZHANG[3]

ABSTRACT

The perfect phylogeny problem is a classical problem in evolutionary tree construction. In this paper, we propose a new model called phylogenetic network with recombination that takes recombination events into account. We show that the problem of finding a perfect phylogenetic network with the minimum number of recombination events is NP-hard; we also present an efficient polynomial time algorithm for an interesting restricted version of the problem.

1. INTRODUCTION

Let S be a set of n species. A *phylogeny* is a rooted tree with n leaves, each of which is uniquely labeled with a species in S. The *perfect phylogeny* problem is a classical problem in evolutionary tree construction. Here each species is described with a set of *characters* and each character has r states that partition S into r equivalence classes. The perfect phylogeny problem asks if there exists a phylogeny T (called *perfect phylogeny*) such that the subgraph of T induced by each equivalence class is connected and, if so, constructs the phylogeny. For simplicity, we only consider binary characters in this paper. That is, $r = 2$ and each character has states 0 and 1.

In a perfect phylogeny, the transformation of states from 0 to 1 occurs at most once for each character. For binary characters, a perfect phylogeny is the most parsimonious tree. If we define the distance between species as the Hamming distance of their character vectors, a perfect phylogeny is just an optimal Steiner tree. Furthermore, such a tree is also *additive*; i.e., the distance between any two species is exactly the same as the length of the path connecting the two species in the tree. (Those facts are not true for characters with more than two states.) See Waterman *et al.* (1977) for more about additive trees. If such a perfect phylogeny exists, the set of characters describing the set of species is *compatible*. Algorithms for testing whether a perfect phylogeny exists are given in Gusfield (1990) and Waterman *et al.* (1977). Day and Sankoff (1986) proposed to find the largest set of characters that can fit a phylogeny. Goldberg *et al.* (1996) proposed a method that uses phylogenetic number to find good phylogenies. Their method restricts the number of times that any given character state arises in the phylogeny.

In this paper, we propose a new approach that takes recombination events into account.

Recombinations

The recombination operation originates from modeling mutations in DNA sequences (Hein, 1990, 1993). Suppose that each species is assigned a sequence. When recombination happens, the ancestral material on

[1]Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong.
[2]Department of Computer Science, University of Western Ontario, London, Ontario N6A 5B7, Canada.
[3]BioInformatics Centre, National University of Singapore, 21 Heng Mui Keng Terrace, Singapore 119613.

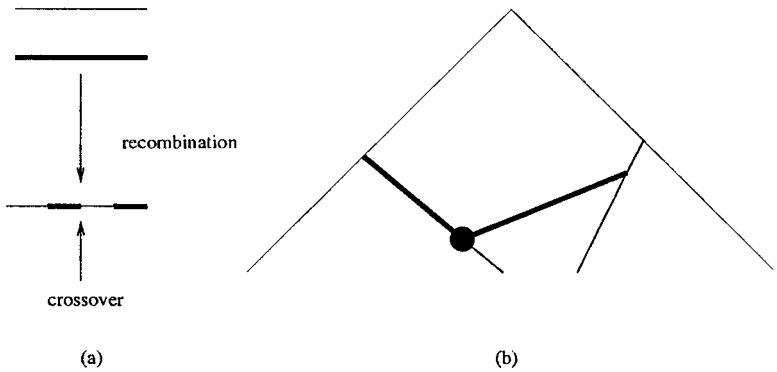

FIG. 1. **(a)** Recombination operation. **(b)** The topology. The dark edges are recombination edges. The circled node is a recombination node.

the present sequence is located on two sequences, one having all the material to the left of a point (called the *recombination point*) and another having all the material to the right of the recombination point. See Fig. 1(a). The recombination point cuts the sequence into two segments.

When recombination occurs, the evolutionary history among a set of species cannot be modeled by a rooted tree. Instead, it can be represented by a rooted *recombination topology* in which some nodes, called *recombination* nodes, have two parents (Hein, 1990, 1993; Hein *et al.*, 1996; Wang *et al.*, to appear). See Fig. 1(b). When using a set of characters to describe species, the linear order among the set of characters may or may not exist. However, the recombination topology remains the same.

Here, we will propose a new model that takes recombination events into account. Similar to the perfect phylogeny problem, we would like to have a recombination topology T such that its restriction on every equivalence class for a character state is connected. Such a recombination topology is called a *perfect phylogenetic network with recombination*. Without loss of generality, we may assume that some given species has state 0 for every character. For each character, the change from 0 to 1 in a recombination topology appears at most once. Thus, we impose that

- (X) the recombination topologies have layouts (the layouts are obtained by organizing the nodes in T level by level) such that, for each character, there is a unique node having state 1 that is the highest and any other nodes having state 1 for this character are its descendants.

Figure 2 gives an example. For a recombination node v, the two edges connecting v to its two parents in the topology are *recombination* edges of v. An edge is a *normal* edge if it is not a recombination edge. A recombination edge *inherits* the character c if both ends of the recombination edge have state 1 for the character c.

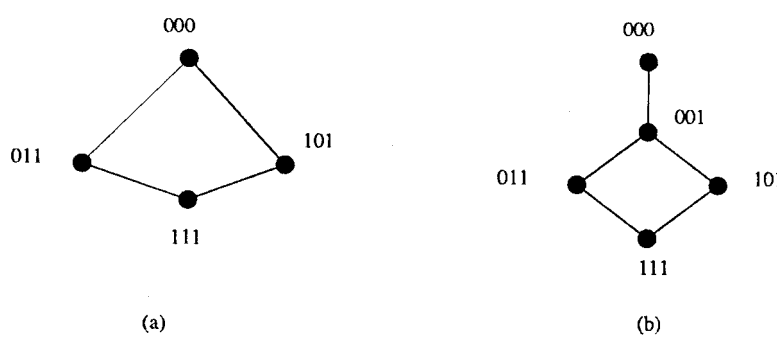

FIG. 2. **(a)** A topology having two highest nodes that have state 1 for a character. **(b)** The alternative topology having a unique highest node that has state 1 for every character.

Obviously, given a set of species S and a set of characters describing the species in S, a perfect phylogenetic network with recombination always exists. Again, using parsimony criteria, we would like to compute a perfect phylogenetic network with recombination that has the smallest number of recombination nodes. The problem defined here is different from the *Steiner trees* problem for vector space, where the distance between species is Hamming distance. The reason is that each recombination node can reduce the cost by more than 1.

In this paper, we study the problem *perfect phylogenetic network with recombination*. Let M be an n by m 0-1 matrix representing n species in terms of m characters that describe the species. $M(i, j)$ has a value 1 if and only if species i has character j. The problem is to find a perfect phylogenetic network with recombination that realizes M and has the smallest number of recombination nodes.

The perfect phylogeny model assumes that a change from state 0 to 1 happens once. If we look at the mutations of DNA or protein sequences, this assumption is certainly too restrictive. However, if we extract the characters from sequences at a high level, e.g., some conserved region exists or not, a gene has a specific character or not, etc, this assumption might be true.

2. NP-COMPLETENESS

In this section, we will show that the perfect phylogenetic network with recombination problem is NP-hard. We first introduce some notation. Let $S' \subseteq S$ be a subset of species and T a tree on the set of species S. $T|S'$ denotes the tree induced from T by deleting the species not in S'. Let $S' \subseteq S$ be a subset of species that are represented by vectors of m elements, one for each character. The lowest common ancestor of S', $lca(S')$, is a vector of m elements such that the j-th element of $lcs(S')$ for character c_j is $\min\{M(i, j) | i \in S'\}$.

Theorem 1. *Perfect phylogenetic network with recombination is NP-hard.*

Proof. The reduction is from *subtree transfer distance between two (rooted) binary trees* that was shown to be NP-hard in [7]. A subtree transfer operation cuts an edge of T and moves the subtree rooted at the lower end of the cut edge to a new branch. Figure 3 gives an example, when T_2 is moved to the branch immediately above T_4. Let T_1 and T_2 be two binary evolutionary trees on a set of species S. The subtree transfer distance between T_1 and T_2 is defined to be the smallest number of subtree transfer operations required to transfer T_1 into T_2.

For a rooted binary tree with n leaves, there are $2(n-1)$ edges. We use $2(n-1)$ characters, each corresponding to an edge, to describe the edges and nodes in the tree. For each edge, there is a character whose state on one end of the edge is 0 and whose state on the other end is 1. The root is assigned a vector containing $2(n-1)$ zeros. Let v be an internal node (possibly the root) and v_1 and v_2 be the two children of v. Let c_1 and c_2 be the two characters corresponding to the edges (v, v_1) and (v, v_2), respectively. Then all the descendant nodes of v_1 (including v_1) in T are assigned 1 for character c_1, and any other nodes in T are assigned 0 for c_1; and all the descendant nodes of v_2 (including v_2) in T are assigned 1 for character c_2, and any other nodes in T are assigned 0 for c_2. (See Fig. 4 for an example.)

Given two rooted binary trees T_1 and T_2 with n leaves (species) each, we use the sets of characters $C_1 = \{c_1, c_2, \ldots, c_{2(n-1)}\}$ and $C_2 = \{c'_1, c'_2, \ldots, c'_{2(n-1)}\}$ to describe T_1 and T_2, respectively. Thus, each leaf in both trees has a vector containing $2(n-1)$ values. Now, we merge the two vectors corresponding to

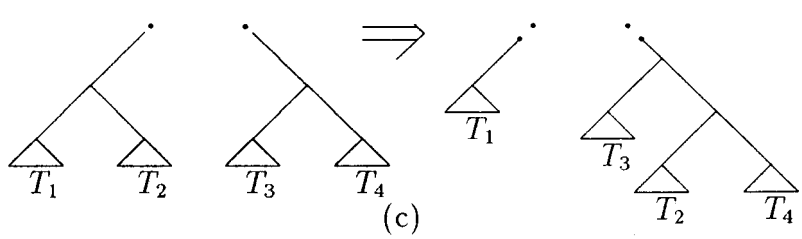

FIG. 3. The subtree transfer operation.

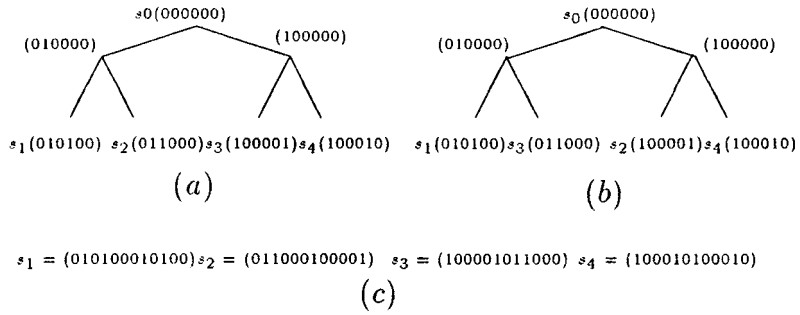

FIG. 4. An example. **(a)** The tree T_1. **(b)** The tree T_2. **(c)** The vectors for species in S.

species s_i in both T_1 and T_2 to get a vector containing $4(n-1)$ values. The instance of perfect phylogenetic network with recombination is then the set of $n+1$ nodes containing the root with vector $(0\ldots0)$; each is described by $4(n-1)$ characters. Figure 1 gives an example.

It is easy to see that, given k subtree transfer operations that modify T_1 into T_2, one can construct a topology with k recombination nodes. (We keep the tree T_1. For each subtree transfer operation that cuts an edge in T_1 and connects the cut subtree to somewhere else in T_1, we simply keep both the cut edge and the new edge.)

Conversely, we want to show that, given a topology with k recombination nodes for the n constructed species, one can find k subtree transfer operations that modify T_1 into T_2.

Claim 2. *Given a topology T with k recombination nodes for the n constructed species, without increasing the number of recombination nodes, one can modify the topology T into T' such that a) each recombination edge in T' inherits characters in either C_1 or C_2; b) one can get T_1 and T_2 simply by deleting one of the recombination edges of every recombination node.*

Proof. We prove it by induction on k, the number of recombination nodes in T. Consider the case, where $k = 1$.

Case 1: The recombination node has no descendant nodes. Then the recombination node represents a species, say, $s' \in S$. (Otherwise, we can simply delete the node.) Deleting the recombination node from T, we get a tree T'' with $n-1$ species.

From the construction, T_1 and T_2 are the unique trees to fit the sets of characters C_1 and C_2. Thus, it is easy to see that $T'' = T_1|(S-\{s'\}) = T_2|(S-\{s'\})$ and the vectors for internal nodes are the lca of the vectors associated with the descendant leaves of the internal nodes.

Now, we can reconnect the species s' to T'' using two edges; one links s' to the original position of s' and T_1, and one links s' to the original position of s' in T_2. Therefore, the claim is true for $k = 1$ when the recombination has no descendant leaves.

Case 2: The recombination node has some descendant leaves. Then the recombination node is the root of a subtree $T_{1,2}$ containing a set of species $S' \subseteq S$, and the two subtrees $T_1|S'$ and $T_2|S'$ are identical to $T_{1,2}$. Thus, we can use $lca(S')$ as the vector associated with the recombination node and treat the recombination node as a species. Therefore, Case 2 degenerates to Case 1.

Assume that the claim is true if there are $k = k'$ recombination nodes in the topology. Consider a topology T with $k'+1$ recombination nodes. From the previous discussion, we can assume that there is a recombination node that does not have any descendant leaf and such a recombination node represents a species, say, s'. Deleting s' from T, we get a topology with k' recombination nodes for $S-\{s\}$. From the assumption, one can get a topology T'' with at most k' recombination nodes such that a) each recombination edge in T' inherits characters in either C_1 or C_2 and b) one can get $T_1|(S-\{s'\})$ and $T_2|(S-\{s'\})$ simply by deleting one of the recombination edges of every recombination node. Now, we can add s' to T'' using one edge to link s' to the original position in T_1 and another edge to link s' to the original position in T_2.

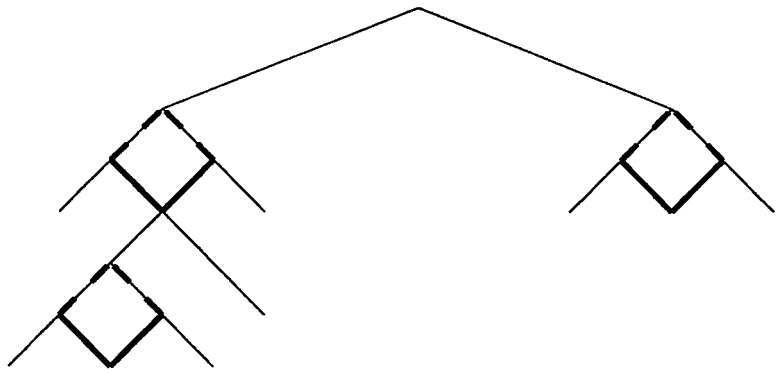

FIG. 5. A topology satisfies (C1). The dark lines are recombination edges. The dashed lines are merged paths for recombination nodes.

From Claim 2, given a topology with k recombination nodes, one can find k recombination operations to transfer T_1 into T_2. This completes the proof. ■

It was shown that the *subtree transfer distance between two binary trees* is hard in terms of approximation; i.e., it is MAX SNP-hard (Hein *et al.*, 1996). In other words, there is no polynomial time approximation scheme (PTAS) for the subtree transfer distance between two binary trees problem unless NP = P. It is easy to see that our reduction is an L-reduction (Hein et al., 1996). Thus, a perfect phylogenetic network with recombination is also MAX SNP-hard.

Theorem 3. *A perfect phylogenetic network with recombination is MAX SNP-hard.*

Theorems 1 and 3 are disappointing. However, in practice, recombinations rarely occur. Thus, it is interesting to consider some restricted versions. The two *merged paths* for a recombination node v are the two paths from the two parents of v to the lowest common ancestor of v's parents in the topology. We will consider the case where

- (C1) in a merged path of a recombination node; there is no node that is in the merged path of a different recombination node.

The condition is also used by Wang *et al.* (to appear). If a topology satisfies (C1), the recombination events are "independent." Figure 5 gives an example.

3. THE RESTRICTED VERSION

In this section, we consider the special case, where (C1) holds. Let c_i be a character and O_i be the set of species in S with $c_i = 1$. If no two columns are identical, then no two O_i's are identical. If $O_i \cap O_j \neq \emptyset$, $O_i \cap O_j \neq O_i$, and $O_i \cap O_j \neq O_j$, then O_i and O_j *conflict* and (O_i, O_j) is called a *conflict* pair. Recall that we assume that the species with all states to be 0's are always in S. Thus, it is easy to see the following.

Lemma 4. *Let c_i and c_j be characters and O_i and O_j be the set of species with $c_i = 1$ and $c_j = 1$, respectively. O_i and O_j conflict if and only if there exist four species such that the states for c_i and c_j are $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$, respectively.*

Proof (only if). The assumption that the root (species) has state 0 for every character implies that there is a species with state 0 for both c_i and c_j. The inequality $O_i \cap O_j \neq \emptyset$ implies that there is a species with state 1 for both c_i and c_j. Similarly, $O_i \cap O_j \neq O_i$ and $O_i \cap O_j \neq O_j$ imply that there are two species with states $(0, 1)$ and $(1, 0)$ for characters c_i and c_i.

The "if" part is obvious. ■

Let O_i and O_j be two conflict pairs. A character $c_{m_{i,j}} \in C$ is a *top* character for O_i and O_j if the following conditions are satisfied: 1) $c_{m_{i,j}}$ does not conflict with any character, 2) $O_i \subseteq O_{m_{i,j}}$ and $O_j \subseteq O_{m_{i,j}}$, and 3) $O_{m_{i,j}}$ is the smallest set satisfying 1) and 2). It is known (Gusfield, 1990) that if two characters c_p and c_q do not conflict, then either O_p and O_q are disjoint or either O_p or O_q contains the other. Thus, $c_{m_{i,j}}$ is well defined and $c_{m_{i,j}}$ is unique for any pair (O_i, O_j).

Theorem 5. *For a given matrix M, a perfect phylogenetic network with recombination satisfying condition (C1) exists if and only if either*

(a) every O_i and O_j do not conflict (in this case, a perfect phylogeny exists); or
(b) if O_i and O_j do conflict,
  (b1) for any $k \notin \{i, j\}$, if O_k and O_i conflict then $O_k \cap O_i = O_i \cap O_j$, and if O_k and O_j conflict then $O_k \cap O_j = O_i \cap O_j$,
  (b2) if two conflict pairs (O_i, O_j) and $(O_{k'}, O_{k''})$ having identical intersections, i.e., $O_i \cap O_j = O_{k''} \cap O_{k''}$, then either
    (1) (O_i, O'_k) and $(O_j, O_{k''})$ are conflict pairs and (O_i, O''_k) and $(O_j, O_{k'})$ are not conflict pairs; or
    (2) (O_i, O''_k) and $(O_j, O_{k'})$ are conflict pairs and (O_i, O'_k) and $(O_j, O_{k''})$ are not conflict pairs,
  (b3) for any conflict pair (O_k, O_l), if $O_k \cap O_l \neq O_i \cap O_j$, then either $O_k \cap O_l$ and $O_i \cap O_j$ are disjoint or one contains the other,
    (b3.1) $O_k \cap O_l$ and $O_i \cap O_j$ are disjoint: then there exist distinct $O_{m_{i,j}}$ and $O_{m_{k,l}}$, i.e., $O_{m_{i,j}} \neq O_{m_{k,l}}$,
    (b3.2) $O_k \cap O_l \supset O_i \cap O_j$: then there exists a character c_m satisfying the following: (1) c_m is not conflict with any character, (2) $O_i \subseteq O_m$ and $O_j \subseteq O_m$, and (3) $O_m \subseteq O_k \cap O_l$.

Proof (if). We prove the lemma by induction on the number of given species. Obviously, the lemma is true for two species. Assume that the lemma is true for l species. Consider the case where there are $l + 1$ species. Suppose that O_i and O_j conflict. From (b3), we can assume that there is no other conflict pair O_k and O_l such that $(O_i \cap O_j) \cap (O_k \cap O_l) \subset O_i \cap O_j$. That is, O_i and O_j is a lowest conflict pair. It is not hard to see that the sets of species $O_i \cap O_j$ and $S - (O_i \cap O_j)$ also satisfy the conditions in the lemma. By assumption, we can construct perfect phylogenetic networks with recombination satisfying condition (C1), say, T_1 and T_2, for $O_i \cap O_j$ and $S - O_i \cap O_j$, respectively. Note that every node of T_1 for $O_i \cap O_j$ has $c_i = 1$ and $c_j = 1$. Moreover, the vector $v(T_1)$ for the root of T_1 is as follows: for any character c, $c = 1$ for $v(T_1)$ if every species in $O_i \cap O_j$ has $c = 1$; otherwise, $c = 0$ for $v(T_1)$.

Now, we want to show that one can connect the root of T_1 to T_2 by using two edges.

From (b2), if (O_{i_1}, O_{j_1}), $(O_{i_2}, O_{j_2}), \ldots, (O_{i_k}, O_{j_k})$ are conflict pairs with identical intersections, i.e., $O_{i_l}, \cap O_{j_l} = O_{i_p}, O_{j_p}$ for any $1 \leq l, p \leq k$, then $\{i_1, i_2, \ldots, i_k, j_1, j_2, \ldots, j_k\}$ can be divided into two disjoint sets A_1 and A_2 such that any pair of characters in A_i does not conflict with each other for $i = 1, 2$. Thus, there exists $s_1 \in S - O_{i_1} \cap O_{j_1}$ and $s_2 \in S - O_{i_1} \cap O_{j_1}$ such that s_1 has state 1 for every character in A_1 and s_2 has state 1 for every character in A_2. (Otherwise, there are a pair of characters c_1 and c_2 in an A_i $(i = 1, 2)$ such that there exist species ss_1 and ss_2 with $(0, 1)$ and $(1, 0)$ as the states for characters c_1 and c_2. Moreover, any species in $O_{i_l} \cap O_{j_l}$ has states $(1, 1)$ for characters c_1 and c_2. Finally, the root of the topology has state $(0, 0)$ for characters c_1 and c_2. From Lemma 4, the pair of characters c_1 and c_2 in A_i conflicts. This contradicts the construction of A_i.) That is, one can find two edges to connect the root of T_1 and T_2 that inherit 1's from T_2 for those characters in A_1 and A_2.

Now, consider those characters not in $A_1 \cup A_2$. Let c_k be a character not in $A_1 \cup A_2$. Since (b1) ensures that (D1) if $c_k = 0$ for $v(T_1)$, then either $c_k = 0$ for s_1 or $c_k = 0$ for s_2. Otherwise, there exist species s_1 and s_2 such that s_1 has $c_i = 0$, $c_j = 1$, and $c_k = 1$ for some $c_i \in A_1$ and $c_j \in A_2$, whereas s_2 has $c_i = 1$, $c_j = 0$, and $c_k = 1$. Note that the root of any topology has state 0 for any character and in T_1 there is a species that has $c_i = 1$, $c_j = 1$, and $c_k = 0$. Thus, from Lemma 4, O_k and O_i are conflict and $O_k \cap O_i \neq O_i \cap O_j$; i.e., (b1) is violated. For any character c_k not in $A_1 \cup A_2$, (D2) if $c_k = 1$ for $v(T_1)$, then either (1) s_i for $i = 1, 2$ has $c_k = 1$ or (2) both s_1 and s_2 have $c_k = 0$. Otherwise, c_k is in conflict with some character in $A_1 \cup A_2$ and thus, according to (b1), c_k should be in $A_1 \cup A_2$. This is a

contradiction. In (1), $v(T_1)$ inherits $c_k = 1$ for some s_i for $i = 1, 2$. In (2), $v(T_1)$ is the first node that changes the state of c_k from 0 to 1. In other words, any species with $c_k = 1$ is in $O_{i_1} \cap O_{j_1}$.

Therefore, one can directly connect the root of T_1 to the two species s_1 and s_2 in T_2. Now, we show that (b3) ensures (C1); i.e., in a merged path of a recombination node there is no node that is in the merged path of another recombination node.

Let O_i and O_j conflict. It is easy to see that the highest node in the two merged paths of the recombination node corresponding to $O_i \cap O_j$ is below the edge where the state of $c_{m_{i,j}}$ changes.

First, we show that the edge, where character $c_{m_{i,j}}$ (the top character for O_i and O_j) changes states, is not on any merged path of any recombination node.

Let O_i and O_j conflict and $c_{m_{i,j}}$ be the top character for O_i and O_j. Let O_k and O_l conflict and $O_k \cap O_l \neq O_i \cap O_j$. We will show that the edge, where character $c_{m_{i,j}}$ changes states, is not on the merged paths for $O_k \cap O_l$. Suppose that the edge, where character $c_{m_{i,j}}$ changes states, is in the merged path from O_k to the common ancestor. Two cases arise:

Case 1: The edge where character c_k changes states is below the edge where character $c_{m_{i,j}}$ changes states. (See Fig. 6(a).) Consider the states of the three characters $c_{m_{i,j}}$, c_k, and c_l. In the path from O_k to the common ancestor, there are two nodes with vectors $(1, 0, 0)$ and $(1, 1, 0)$, where the first component is for $c_{m_{i,j}}$, the second component is for c_k, and the third is for c_l, respectively. In the path from O_l to the common ancestor, there is a node with vector $(0, 0, 1)$. The common ancestor has the vector $(0, 0, 0)$. Now, consider the vector for the recombination node; the vector could be either $(1, 1, 1)$ or $(0, 1, 1)$. In the former case, $c_{m_{i,j}}$ is in conflict with O_l, and in the later case, $c_{m_{i,j}}$ is in conflict with O_k. Since $c_{m_{i,j}}$ is not in conflict with any character, we know that the assumption is not true. That is, the edge, where character $c_{m_{i,j}}$ changes states, is not on the merged paths for $O_k \cap O_l$.

Case 2: The edge where character $c_{m_{i,j}}$ changes states is below the edge where character c_k changes states. (See Fig. 6(b).) In this case, the vector for the recombination node must be $(0, 1, 1)$. (Otherwise, $c_{m_{i,j}}$ is conflict with c_l.) Thus, we can reconstruct the topology such that the edge where character $c_{m_{i,j}}$ changes states is not in the merged path. (See Fig. 6(c).)

Now, we have to consider the two cases, where either $O_i \cap O_j$ and $O_k \cap O_l$ are disjoint or one contains the other.

From (b3.1), the merged paths of O_i and O_j do not share any node with the merged paths of O_k and O_l.

From (b3.2) (3), $O_{m_{i,j}} \subseteq O_k \cap O_l$, we know that the edge where the state of $c_{m_{i,j}}$ changes is below the recombination node corresponding to $O_k \cap O_l$. Thus, we can conclude that (C1) holds.

Proof (continued; only if). If there exists a perfect phylogenetic network satisfying (C1), then it is easy to verify that the conditions in Lemma 5 hold. ■

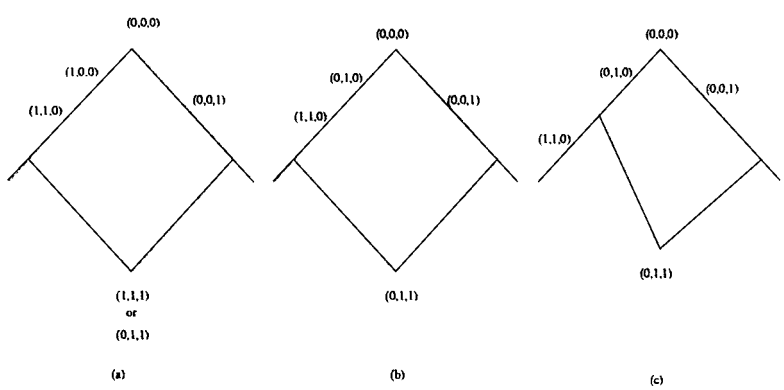

FIG. 6. The edge, where character $c_{m_{i,j}}$ changes states, is not on the merged paths for $O_k \cap O_l$. The first component is for $c_{m_{i,j}}$, the second component is for c_k, and the third is for c_l.

Theorem 6. *If a perfect phylogenetic network with recombination satisfying (C1) exists, then such a topology has the minimum number of recombination nodes.*

Proof. From (b1), (b2), and (b3), it is easy to see that the number of distinct $O_i \cap O_j$'s are fixed. Obviously, each $O_i \cap O_j$ corresponds to at least one recombination node. From the proof of Theorem 5, we know that one distinct $O_i \cap O_j$ does contribute exactly one recombination node if a perfect phylogenetic network with recombination satisfying (C1) exists. Therefore, the topology has the minimum number of recombination nodes. ■

3.1. The algorithm

We give an algorithm to test whether a perfect phylogenetic network with recombination satisfying (C1) exists and construct the phylogenetic network if such a perfect phylogenetic network exists.

First, we preprocess the matrix M. Considering each column of M as a binary number with the most significant bit in row 1, we sort these numbers into decreasing order and place the largest number in column 1. Delete any column that is identical to the column on its right. The new matrix is denoted as M'. The preprocessing needs $O(nm)$ time, where n is the number of species and m is the number of columns in M.

Lemma 7. *If there is a perfect phylogenetic network with recombination satisfying (C1) for a matrix M and M does not contain identical columns, then there are $O(n)$ columns in M.*

Proof. If a perfect phylogenetic network with recombination satisfying (C1) exists, then the number of normal edges in the topology is less than $2n$. (Delete all the recombination edges and count the remaining edges in the set of subtrees.) Since each character corresponds to at most one edge, there are at most $2n$ characters and thus at most $2n$ columns in M. ■

By Lemma 7, the size of M' is at most $O(n^2)$. Obviously, there is a perfect phylogenetic network with recombination satisfying (C1) for M if and only if there is a perfect phylogenetic network with recombination satisfying (C1) for M'.

Now, we can use the algorithm testCondition to test whether a perfect phylogenetic network with recombination satisfying (C1) exists for a given preprocessed matrix M'. The algorithm is given in Fig. 7.

Theorem 8. *The algorithm testCondition needs $O(n^4)$ time.*

Proof. Step 1 tests the condition (a) and needs $O(n^2)$ time. Step 2 tests conditions (b1) and (b2). Using a trivial implementation, Step 2 needs $O(n^4)$ time. Testing (b3.1) and (b3.2) for two distinct conflict pairs needs $O(n^2)$ time. From Lemma 7, there are at most $O(n^2)$ pairs of conflict pairs. Thus, $O(n^4)$ time is required to test (b3). Therefore, the total time required is $O(n^4)$. ■

If there does not exist a perfect phylogeny and testCondition outputs "yes," one can construct a topology as follows:

1. Construct perfect phylogenies for those $O_i \cap O_j$'s not contained in any other $O_k \cap O_j$'s.
2. Ignore those species used in step 1 and repeat step 1 for other $O_i \cap O_j$'s until no conflict pair exists.
3. Connect those perfect phylogenies using recombination edges.

From the proof of Theorem 5, we know that we can always successfully connect two topologies if all the conditions in Theorem 5 hold. A vector v *covers* a vector u if, for any elements $v(i)$ of v and $u(i)$ of u, $v(i) = 1$ implies $u(i) = 1$. To connect topologies, we can simply look for two nodes that are as low as possible in the constructed topology such that their vectors cover the vector of the root for the other topology. Then we use two edges to connect the two topologies. This needs $O(n^2)$ time. Since we can have at most $O(n)$ perfect phylogenies in total, the total time required to connect all those topologies is $O(n^3)$.

Algorithm testCondition

1. use Gusfield's algorithm [3] to test if a perfect phylogeny exists.
2. **for** each columns i in M' **do**

   find all columns that are conflict with column i

   let J be the set of indexes of columns that are conflict with column i

   test if $O_i \cap O_{j'} = O_i \cap O_l$ for every $l \in J$,

   where column j' is the rightmost column in M' such that $j' \in J$.

   if not, output "(C1) does not hold".

   find all columns that are conflict with column j for every $j \in J$.

   let K be the set of indexes of columns that are conflict with column j for every $j \in J$.

   if $O_k \cap O_l \neq O_i \cap O_j$ for some $l \in J$ and $k \in K$ **then**

   output "(C1) does not hold".

   if some pair of columns in J or K is conflict **then**

   output "(C1) does not hold".

3. test condition (b3)
4. **if** Steps 1, and 2 are all successful **then**

   output "yes"

   else output "no"

FIG. 7. The algorithm to test if a perfect phylogenetic network with recombination satisfying (C1) exists.

Constructing a perfect phylogeny needs $O(n_i^2)$ time if one uses Gusfield's algorithm (1990), where n_i is the number of species contained in the perfect phylogeny. Thus, the total time required to construct all small perfect phylogenies is $\sum_{i=1}^{k} O(n_i^2)$, where $\sum_{i=1}^{k} n_i = n$. Obviously, $\sum_{i=1}^{k} O(n_i^2)$ is $O(n^2)$. Therefore, we need at most $O(n^3)$ time to construct the phylogenetic network.

Remarks. It is interesting to give an approximation algorithm with some guaranteed performance ratio for the general case. It is not known whether the problem has a constant ratio. It seems that the ideas for the approximation algorithm of subtree transfer distance (Hein *et al.*, 1996) do not work here.

ACKNOWLEDGMENT

We thank the referee for helpful suggestions. Lusheng Wang is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. 9040444].

REFERENCES

DasGupta, B., He, X., Jiang, T., Li, M., Tromp, J., and Zhang, L. 1997. On distances between phylogenetic trees *in Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1997, New Orleans.

Day, W.H.E., and Sankoff, D. 1986. Computational complexity of inferring phylogenies by compatibility. *Syst. Zool.* 35, 224–229.

Gusfield, D. 1990. Efficient algorithms for inferring evolutionary trees. *Networks* 21, 19–28.

Goldberg, L.A., Goldberg, P.W., Phillips, C.A., Sweedyk, E., Warnow, T. 1996. Minimizing phylogenetic number to find good evolutionary tree. *Discrete Appl. Math.* 71, 111–136.

Hein, J. 1990. Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.* 98, 185–200.

Hein, J. 1993. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.* 36, 396–405.

Hein, J., Jiang, T., Wang, L., and Zhang, K. 1996. On the complexity of comparing evolutionary trees. *Discrete Appl. Math.* 71, 153–169.

Stahl, F.W. 1987. Genetic recombination. *Sci. Am.*, February, 90–101.

Wang, L., Ma, B., and Li, M. Tree alignment with recombination. *Discrete Appl. Math.* To appear.

Watson, J.D., Hopkins, N.H., Roberts, J.W., Steitz, J.A., Weiner, A.M. 1987. *Molecular Biology of the Gene*, 4th ed., Benjamin-Cummings, Menlo Park, CA.

Waterman, M.S., Smith, T.F., Singh, M., and Beyer, W.A. 1977. Additive evolutionary trees. *J. Theoret. Biol.* 64, 199–213.

Address correspondence to:
Lusheng Wang
Department of Computer Science
City University of Hong Kong
83 Tat Chee Avenue
Kowloon, Hong Kong

E-mail: lwang@cs.cityu.edu.hk