Perfect Phylogenetic Networks with Recombination *

Lusheng Wang
Dept. of Computer Sci.
City Univ. of Hong Kong
83 Tat Chee Avenue
Hong Kong
lwang@cs.cityu.edu.hk

Kaizhong Zhang
Dept. of Computer Sci.
Univ. of Western Ontario
London, Ont. N6A 5B7
Canada
kzhang@csd.uwo.ca

Louxin Zhang
BioInformatics Centre
Nat. Univ. of Singapore
21 Heng Mui Keng Terrace
Singapore 119613
lxzhang@bic.nus.edu.sg

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, construct 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 are not true for characters with more than two states.) See [8] 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 [3, 8]. However, a perfect phylogeny rarely exists for real data. When a perfect phylogeny does not exist, Day and Sankoff [1] proposed to find the largest set of characters that can fit a phylogeny. Unfortunately, the problem is NP-hard. Goldberg et. al proposed a method that use phylogenetic number to find good phylogenies. Their method restricts the number of times that any given character state arises in the phylogeny [2].

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 [4, 5]. Suppose that each species is assigned a sequence. When recombination happens, the ancestral material on 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 Figure 1 (a). The recombination point cuts the sequence into two segments.

When recombination occurs, the evolutionary history among a set of species can not 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 [4, 5, 6, 7]. (See Figure 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 phy-*

*(Produces the permission block, copyright information and page numbering). For use with ACM_PROC_ARTICLE-SP.CLS V2.0. Supported by ACM.

SAC 2001, Las Vegas, NV

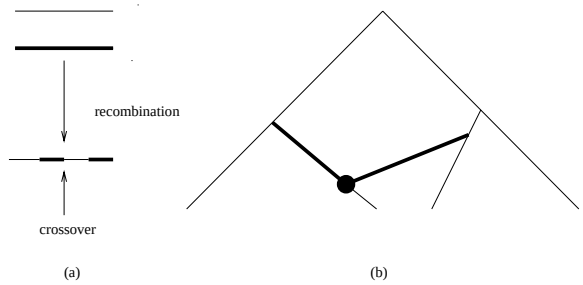

Figure 1: (a) Recombination operation. (b) The topology. The dark edges are reco mbination edges. The circled node is a recombination node.

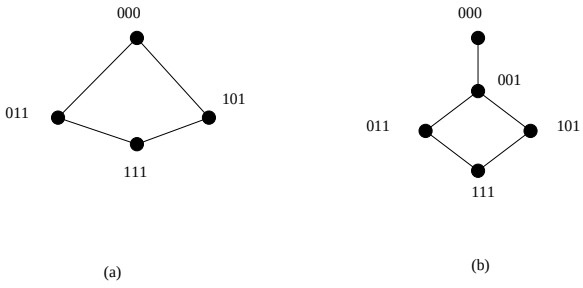

Figure 2: (a) a topology having two highest nodes that have state 1 for a character. (b) the alternative topology having a unique highest node that have state 1 for every character.

logenetic 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

- 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.

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 a 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.

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. $lca(S')$, the lowest common ancestor of 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.*

THEOREM 2. *Perfect phylogenetic network with recombination is MAX SNP-hard.*

Theorems 1 and 2 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 in [7]. If a topology satisfies (C1), the recombination events are "independent". Figure 3 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 are *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 that

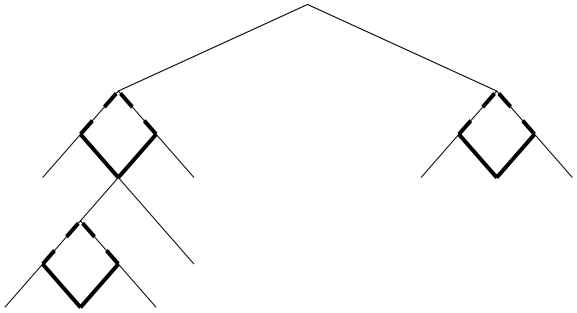

Figure 3: A topology satisfying (C1). The dark lines are recombination edges. The dashed lines are merged paths for recombination nodes.

LEMMA 3. *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 are 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.*

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) $c_{m_{i,j}}$ is 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 [3] that if two characters c_p and c_q are not conflict, then either O_p and O_q are disjoint or one of O_p and 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 4. *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 are not conflict (in this case, a perfect phylogeny exists); or*
- *(b) if O_i and O_j are conflict,*
  - *(b1) for any $k \notin \{i, j\}$, if O_k and O_i are conflict then $O_k \cap O_i = O_i \cap O_j$ and if O_k and O_j are conflict then $O_k \cap O_j = O_i \cap O_j$.*
  - *(b2) if two conflict pairs (O_i, O_j) and $(O_{k'}, O_{k''})$ have 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 exist a character c_m satisfying (1) c_m is not conflict with any character and (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 are 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}), ..., (O_{i_k}, O_{j_k}) are conflict pairs with identical intersections, i.e., $O_{i_l} \cap O_{j_l} = O_{i_p} \cap O_{j_p}$ 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 is 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 a 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 3, the pair of characters c_1 and c_2 in A_i is conflict. This contradicts the construction of A_i.) That is, one can find two edges to connect the root of T_1 with 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 $A_1 \cup A_2$. (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 has $c_i = 1$, $c_j = 1$ and $c_k = 0$. Thus, from Lemma 3 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 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 be 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 be conflict and $c_{m_{i,j}}$ be the top character for O_i and O_j. Let O_k and O_l be 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 Figure 4 (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 conflict with O_l and in the later case, $c_{m_{i,j}}$ is conflict with O_k. Since $c_{m_{i,j}}$ is not 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 Figure 4 (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 Figure 4 (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.

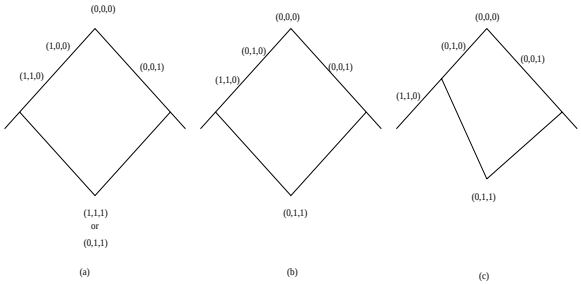

Figure 4: The edge, where character $c_{m_{i,j}}$ changes states, is not on the merged paths for $O_k \cap O_l$. The first component is for $c_{m_{i,j}}$, the second component is for c_k and the third is for c_l, respectively.

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 is below the recombination node corresponding to $O_k \cap O_l$. Thus, we can conclude that (C1) holds.

(only if) If there exists a perfect phylogenetic network satisfying (C1), then it is easy to verify that the conditions in Lemma 4 hold. □

THEOREM 5. *If a perfect phylogenetic network with recombination satisfying (C1) exists, then such a topology has the minimum number of recombination nodes.*

The algorithm

We give an algorithm to test if 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 preprocess needs $O(nm)$ time, where n is the number of species and m is the number of columns in M.

LEMMA 6. *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.*

By Lemma 6, 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

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"

Figure 5: The algorithm to test if a perfect phylogenetic network with recombination satisfying (C1) exists.

is a perfect phylogenetic network with recombination satisfying (C1) for M'.

Now, we can use algorithm testCondition to test if a perfect phylogenetic network with recombination satisfying (C1) exists for a given preprocessed matrix M'. The algorithm is given in Figure5.

THEOREM 7. *testCondition needs $O(n^4)$ time.*

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 4, we know that we can always successfully connect two topologies if all the conditions in Theorem 4 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 $O(n^3)$.

Constructing a perfect phylogeny needs $O(n_i^2)$ time if one use Gusfield's algorithm [3], 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 [6] do not work here.

4. REFERENCES

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

[2] L. A. Goldberg, P. W. Goldberg, C. A. Phillips, E. Sweedyk, and T.Warnow. Minimizing phylogenetic number to find good evolutionary tree. *Discrete Applied Mathematics*, 71:111–136, 1996.

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

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

[5] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evo.*, 36:396–405, 1993.

[6] J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, 1996.

[7] L. Wang, B. Ma, and M. Li. Tree alignment with recombination. *Discrete Applied Mathematics.*

[8] M. Waterman, T. F. Smith, M. Singh, and W. A. Beyer. Additive evolutionary trees. *Journal of Theoretical Biology*, 64:199–213, 1977.