

Reconstructing Recombination Network from Sequence Data: The Small Parsimony Problem

C. Thach Nguyen, Nguyen Bao Nguyen, Wing-Kin Sung, and Louxin Zhang

Abstract—The small parsimony problem is studied for reconstructing recombination networks from sequence data. The small parsimony problem is polynomial-time solvable for phylogenetic trees. However, the problem is proven NP-hard even for galled recombination networks. A dynamic programming algorithm is also developed to solve the small parsimony problem. It takes $O(dn2^{3h})$ time on an input recombination network over length- d sequences in which there are h recombination and $n - h$ tree nodes.

Index Terms—Recombination network, phylogenetic network, parsimony method, NP-hardness, approximability, dynamic programming.

1 INTRODUCTION

ANALYSIS of genomic sequences indicates that horizontal gene transfer (HGT), recombination, and other non-point mutations often occur in genomic evolution. HGT occurs when genetic material transfers from a species to another remotely related species. HGT events are common in the bacterial and other prokaryotic genomes [16], [3], [20]. New evidence also indicates that HGT events might occur in eukaryotic genomes [10].

Recombination is another important mutational event occurring in most genomes. It allows different species to share common genetic materials. A species is defined as an interbreeding group of organisms that are capable of producing fertile offspring. Within a species, genetic material is often exchanged due to high rate gene flow and meiotic recombination. Meiotic recombination takes two equal length DNA segments and produces a third one of the same length by concatenating a prefix of one sequence and a suffix of the other. Studying recombination is central to modern genetics.

As genomes evolved both vertically and horizontally, acyclic directed networks (i.e., trees with reticulation branches) are considered to be more suitable for modeling genomic evolution. Such a model is called a *phylogenetic network*. The *recombination network* is a special type of the phylogenetic network in which each reticulation node corresponds to a recombination operation. Reconstructing phylogenetic networks has been extensively studied recently (see, for example, survey articles [1], [17]). Phylogenetic networks can be reconstructed from a set of gene trees [11], [12], [14], [15], [19] or sequences [5], [6], [7], [9], [13],

[18], [22]. In this paper, we focus on the parsimony approach in reconstructing recombination networks from sequence data.

In phylogeny reconstruction, the parsimony approach is to find a phylogenetic tree that requests the minimum number of mutation events (called the parsimony score) to explain the given sequence data. It generalizes to phylogenetic network reconstruction by considering the set of (directed) subtrees of a network which have the same set of leaves as the network. Consider a phylogenetic network \mathcal{N} in which all nodes are labeled with sequences of equal length d . Let $Trees_{\mathcal{N}}$ be the set of directed subtrees of \mathcal{N} with the same set of leaves. For any site $1 \leq i \leq d$ in the sequences, each subtree $T \in Trees_{\mathcal{N}}$ gives a parsimony score $bs_p(T, i)$. The parsimony score $bs_p(\mathcal{N})$ of the network \mathcal{N} is defined as

$$bs_p(\mathcal{N}) = \sum_{1 \leq i \leq d} \min_{T \in Trees_{\mathcal{N}}} s_p(T, i)$$

[8], [9], see also Section 3.3 for details). Given a set of sequences of equal length, the parsimony approach is to find a phylogenetic network that contains n leaves labeled by the given sequences and has the minimum parsimony score over all such networks.

Since a phylogenetic tree is a network without reticulation nodes, reconstructing phylogenetic network with the given number of reticulation nodes is NP-hard. Hence, heuristic methods have been developed for the purpose [9], [18]. An important subproblem arising from such a reconstructing task is how to label the internal nodes of a network over the given sequences to achieve the minimum parsimony score. Such a problem is called the *small parsimony problem*. It is well known that the small parsimony problem is polynomial-time solvable for phylogenetic trees [4]. Hence, it is interesting and important to investigate whether it is polynomial-time solvable or not for phylogenetic networks [18].

In this paper, we study the small parsimony problem for reconstructing recombination networks. After formalizing the small parsimony problem for recombination networks in Section 2, we prove that it is NP-hard even for galled

• C.T. Nguyen and L. Zhang are with the Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543. E-mail: {thachnguyen, matzlx}@nus.edu.sg.

• N.B. Nguyen and W.-K. Sung are with the Department of Computer Science, National University of Singapore, 3 Science Drive 2, Singapore 117543. E-mail: {baonguyen, dcswwks}@nus.edu.sg.

Manuscript received 17 Mar. 2006; revised 1 June 2006; accepted 24 Aug. 2006; published online 10 Jan. 2007.

For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0082-0306. Digital Object Identifier no. 10.1109/TCBB.2007.1018.

networks in Section 3. We also generalize such an NP-hard proof to the phylogenetic networks in which the type of each reticulation node is unspecified. Since the parsimony score is defined differently for general phylogenetic networks, such a generalization is not trivial. In Section 4, we present a dynamic programming algorithm for the small parsimony problem, which takes $O(dn2^{3h})$ time on an input recombination network with h recombination and $n - h$ tree nodes, and leaves labeled by length- d sequences.

2 BASIC CONCEPTS AND RESULTS

2.1 Recombination Networks

Recall that a phylogenetic tree over a set of sequences is a binary, unordered, (directed) rooted tree whose leaves are one-to-one labeled with the given sequences. In a phylogenetic tree, each edge is directed from the endpoint closer to the root to the other endpoint; each internal node is implicitly labeled with a sequence.

A recombination network can be seen as a generalization of a phylogenetic tree. Formally, a recombination network is a connected, acyclic directed graph having the following properties:

1. It contains a unique node (called the *root*) with no incoming edges, a set of *internal* nodes that have both incoming and outgoing edges, and a set of nodes (called the *leaves*) with no outgoing edge. Each internal node has one or two incoming edges. A node with one incoming edge has two outgoing edges and is called a *tree node*; a node with two incoming edges has one outgoing node and is called a *recombination node*. An edge is called a *tree edge* if it enters a tree node; it is called a *recombination edge* if it enters a recombination node.
2. There are exactly two recombination edges entering each recombination node. These two edges are labeled with p and s , respectively. In addition, a site is associated with a recombination node, called the *crossover position*.
3. All nodes are labeled with sequences of equal length d . In particular, leaves are one-to-one labeled with the given sequences. For a recombination node v with crossover position i ($1 \leq i \leq d + 1$), let e and e' be the p -edge and s -edge entering v from w and u , respectively. The first $i - 1$ sites of v 's label are inherited from w and the last $d - i + 1$ sites are inherited from u .

In the topology part of an ancestral recombination graph (see, for example, [21], [5]), for each tree edge (u, v) , the labels of u and v are different in at most one site. Here, we relax the condition. In our model, the labels of the two endpoints of a tree edge can be different on several sites.

In the reconstruction of phylogenetic networks, galled networks are particularly interesting [11], [5], [7]. In a recombination network, for each recombination node x , there are two node-disjoint directed paths that come out of some node y and meet at x . These two paths together define a *recombination cycle*. A recombination cycle is called a *gall* if it shares no node with any other recombination cycle. A phylogenetic network is called a *galled network* if every

recombination cycle is a gall. In [22], Wang et al. first proposed studying the problem of determining if a set of sequences can be derived from a galled network or not.

2.2 Small Parsimony Problem

Let \mathcal{N} be a recombination network with nodes labeled with length- d sequences. Assume that \mathcal{N} contains h recombination nodes r_1, r_2, \dots, r_h that have crossover positions c_1, c_2, \dots, c_h , respectively. For each i ($1 \leq i \leq d$), \mathcal{N} induces a unique tree T_i : It contains the p -edge of r_j if $i < c_j$ and the s -edge of r_j if $i \geq c_j$ for each recombination node r_j . The leaves of \mathcal{N} are obviously the leaves of T_i . However, T_i may have some recombination nodes as its leaves. It may also contain nonroot degree-2 nodes. Therefore, in the following definition of parsimony score, we only consider the edges that are on the paths from the root to the network leaves. The *parsimony score* of the network \mathcal{N} is defined as¹

$$s_p(\mathcal{N}, l) = \sum_{1 \leq i \leq d} \sum_{(u,v) \in T_i} h(l(u)[i], l(v)[i]),$$

where $h(\cdot)$ is the Hamming distance function, $l(u)$ is the label of an internal node, and $l(u)[i]$ is the i th character of $l(u)$.

The *small parsimony problem* is: Given a recombination network topology with leaves labeled with equal-length sequences, label the internal nodes with the sequence of the same length so that the resulting network has the minimum parsimony score.

Given a recombination network topology G over a set of length- d sequences. If the p -edge, the s -edge, and the crossover position c for every recombination node are known, then, for each site i , T_i in the parsimony score definition is uniquely determined. Therefore, by applying the Fitch algorithm to each T_i , we are able to label the internal nodes of G site by site so that the resulting network has the minimum parsimony score. This proves that:

Theorem 2.1. *The small parsimony problem is polynomial-time solvable if the recombination nodes are completely specified in the given topology.*

Hence, in the rest of this paper, we shall study the small parsimony problem for the recombination networks in which the recombination nodes are not completely specified:

Given a recombination network topology \mathcal{N} with leaves labeled by length- d sequences. Assign the p and s -edges and crossover position for each recombination node and label the internal nodes with length- d sequences so that the resulting network \mathcal{N} has the smallest parsimony score.

1. The recombination networks studied in this paper are so-called single-crossover recombination networks [6], [7]. In a general recombination network, a recombination node can be multicrossover, where more than one segment of its sequences is inherited from either of its parents. Obviously, a multicrossover recombination node can be specified by a set of crossover positions (i_1, i_2, \dots, i_k) such that $1 \leq i_1 < i_2 < \dots < i_k \leq d + 1$, where d is the length of the sequence at the node. It is not hard to see that the parsimony score can be similarly defined for the multicrossover recombination networks.

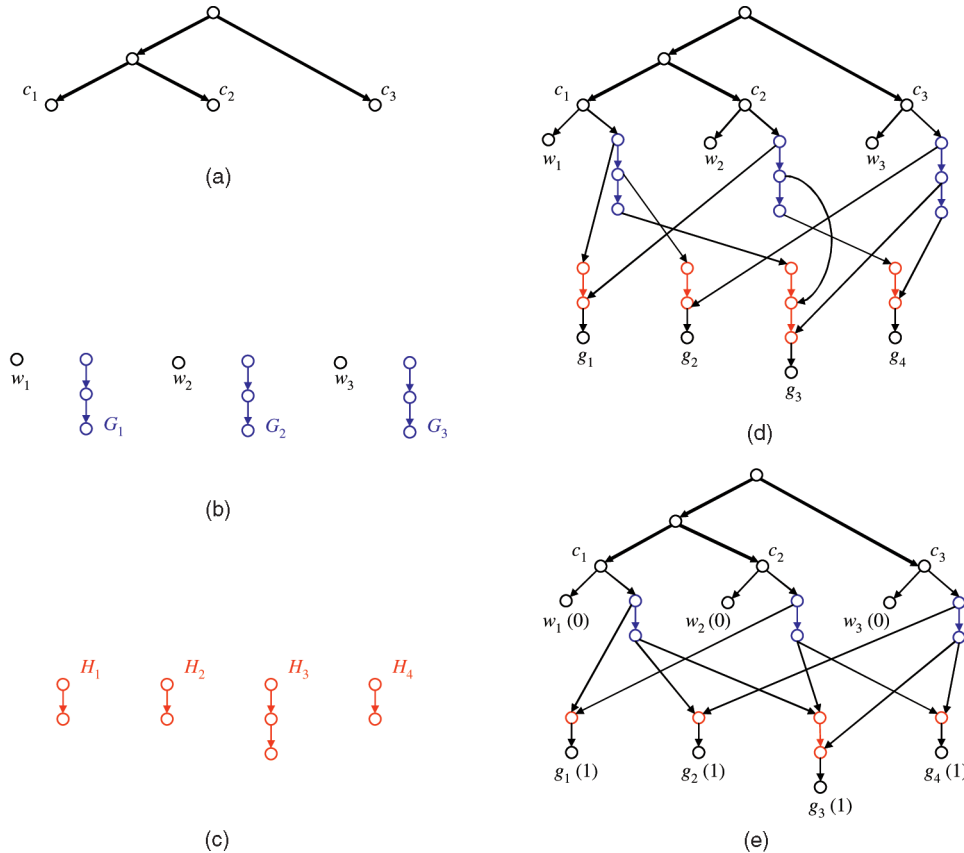


Fig. 1. Four steps to construct a network corresponding to the instance $(S = \{s_1, s_2, s_3, s_4\}, \mathcal{C} = \{C_1 = \{s_1, s_2, s_3\}, C_2 = \{s_1, s_3, s_4\}, C_3 = \{s_2, s_3, s_4\}\})$ of the SET COVER problem.

3 NP-HARDNESS

3.1 A Simple Proof

Theorem 3.1. *The small parsimony problem is NP-hard for the recombination networks in which the recombination nodes are not fully specified.*

Proof. We prove the result by a reduction from the SET COVER problem: Given a finite set S and a subset collection \mathcal{C} that covers S , i.e., $\bigcup_{C \in \mathcal{C}} C = S$, find a smallest subcollection of \mathcal{C} covering S . Given an instance (S, \mathcal{C}) of the SET COVER problem, a finite set $S = \{s_1, s_2, \dots, s_n\}$ and a subset collection $\mathcal{C} = \{C_1, C_2, \dots, C_m \mid C_i \subseteq S\}$ such that $\bigcup_{1 \leq i \leq m} C_i = S$, we construct a recombination network topology \mathcal{N} from (S, \mathcal{C}) in four steps as illustrated in Fig. 1:

1. Choose a binary tree BT of m leaves c_1, c_2, \dots, c_m which correspond one-to-one to the subsets C_1, C_2, \dots, C_m in the collection \mathcal{C} (Fig. 1a).
2. For each subset $C_i \in \mathcal{C}$ with k elements, construct a node w_i and a directed line graph G_i of k vertices, each corresponding to an element in C_i (Fig. 1b).
3. For each element $s_j \in S$, if it is contained in k' subsets in \mathcal{C} , construct a node g_j and a directed line graph H_j of k' elements, each corresponding to a subset that contains s_j (Fig. 1c).
4. Finally, for each element $s_j \in S$ and each subset $C_i \in \mathcal{C}$ such that $s_j \in C_i$, we add an edge from the

t th node in G_i to the t' th node in H_j if s_j is the t th element in C_i and C_i is the t' th subset containing s_j . When $t = |C_i|$ or $t' = 1$, the procedure will produce degree-2 nodes (Fig. 1d). Such degree-2 nodes will be contracted. Moreover, we add an edge from c_i to w_i , from c_i to the first node in G_i for each i , and from the last node in H_j to g_j for each j (Fig. 1e).

The resulting network \mathcal{N} has $m + n$ leaves: $w_1, w_2, \dots, w_m, g_1, g_2, \dots, g_n$. We label each w_i with the single-bit string "0" and each g_j by the single-bit string "1." In other words, there is only one site in the network. In addition, $k' - 1$ recombination nodes are formed from each s_j if it is contained in k' subsets. For each recombination node, we will label its recombination edges with p and s -edges in a specific way.

Let $c(S, \mathcal{C})$ denote the number of subsets in the minimum cover of S for the instance (S, \mathcal{C}) and $s_p(\mathcal{N})$ the parsimony score of the recombination network \mathcal{N} . We prove that $c(S, \mathcal{C}) = s_p(\mathcal{N})$. Let $\mathcal{C}' \subset \mathcal{C}$ be the cover of size $c(S, \mathcal{C})$ for S . Then, for every $s_i \in S$, we choose a "representative" subset $C_{rep(i)} \in \mathcal{C}'$ that contains s_i . Consider the subtree T of \mathcal{N} that is composed of 1) the binary tree BT —the top part of \mathcal{N} , 2) the edges to w_1, w_2, \dots, w_m , and 3) the unique path from $c_{rep(i)}$ to the leaf g_i . Fig. 2 shows such a tree when $rep(1) = rep(2) = 1$ and $rep(3) = rep(4) = 2$ for the set cover example illustrated in Fig. 1.

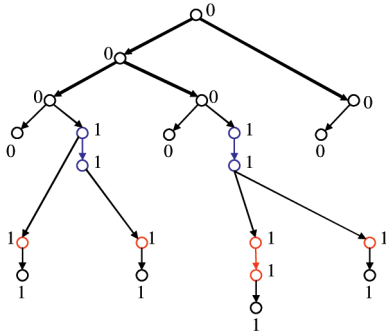


Fig. 2. The subtree of \mathcal{N} in Fig. 1 when $rep(1) = rep(2) = 1$ and $rep(3) = rep(4) = 2$.

We label all nodes in BT , including c_i s, with string 0 and all of the other internal nodes with string 1. For each recombination node r contained in the tree T , we set its crossover position to 2 if the p -edge entering r is in T and to 1 otherwise. For other recombination nodes not contained in T , we set their crossover position arbitrarily. Under this assignment, T is one candidate binary tree for computing the parsimony score for \mathcal{N} . Let $l(u)$ denote the label of u for each node u in the network. By construction, for an edge $e = (u, v)$, $h(l(u), l(v)) = 1$ if and only if $u = c_{rep(i)}$ and $v \neq w_i$ for some i . Hence, the score of T is equal to the number of subsets in \mathcal{C}' and the parsimony score $s_p(\mathcal{N}) \leq c(S, \mathcal{C})$.

Conversely, consider a labeling of \mathcal{N} and a binary tree T contained in \mathcal{N} such that $s_p(T) = s_p(\mathcal{N})$. By construction, if there is a path from c_i to g_j in T , then C_i must contain s_j and, for each g_j , the path from the root to g_j must pass through a c_i . Hence,

$$\mathcal{C}' = \{C_i \in \mathcal{C} | \text{there exists a path from } c_i \text{ to } g_k \text{ for some } k\}$$

is a cover of S . Since w_i and g_i are labeled differently, we have $s_p(\mathcal{N}) = s_p(T) \geq |\mathcal{C}'| \geq c(S, \mathcal{C})$. This concludes the proof. \square

Remark. 1) The NP-hardness result in Theorem 3.1 follows from the fact that the single-site sequence of each recombination node r can evolve from either of the two recombination edges entering r in the network \mathcal{N} constructed in the proof. Assume the p and s -edges are specified at r . If the crossover position at r is set to 2, then the sequence of r is forced to evolve along the p -edge; if the crossover position at r is set to 1, the sequence of r is forced to evolve along the s -edge. Hence, the small

parsimony problem remains NP-hard when the recombination edges are specified, but the crossover position is not known for each recombination node.

Assume the crossover position at r is set to 2; if we assign a recombination edge entering r as the p -edge, then the sequence of r is evolved along this edge; otherwise, the sequence of r is evolved along the other edge. Therefore, the small parsimony problem also remains NP-hard when the crossover position is known, but the recombination edges are not specified for each recombination node.

2) The proof of Theorem 3.1 actually implies that approximating the parsimony score within ratio $c \log n$ is NP-hard for some constant c since the SET COVER problem cannot be approximated with ratio $c \log n$.

3.2 NP-Hardness for Galled Networks

In this section, we show that the small parsimony problem remains NP-hard for galled recombination networks. However, the proof is more sophisticated. It follows a reduction from the VERTEX COVER problem. Recall that the VERTEX COVER problem is: Given a graph G and a number k , determine if there is a k -vertex cover V' of G , i.e., a subset V' of $V(G)$ such that $|V'| = k$ and, for every edge $(u, v) \in E(G)$, either $u \in V'$ or $v \in V'$.

Given a graph G with n vertices $u_1, u_2, u_3, \dots, u_n$ and m edges, we construct a corresponding network topology \mathcal{N} in three steps:

1. For each edge $e = (u_i, u_j) \in E(G)$, construct an edge network C_e as in Fig. 3a. Here, when we mention an edge $e = (u_i, u_j)$, we assume that $i < j$.
2. Each edge network is extended by adding two more vertices, $P(e)$ and $Q(e)$, in such a way that, in the resulting extended network, $Q(e)$ and the root of C_e are the children of $P(e)$.
3. Construct an arbitrary binary tree with m leaves to connect all of the extended edge networks together, as in Fig. 3b, where each triangle represents an edge network defined in (1).

For each edge $e = (u_i, u_j)$ ($i < j$), we label the leaves of the edge subnetwork C_e by length- $(n + 2)$ sequences. All of these sequence labels are only different in four positions, 1, 2, $i + 2$, $j + 2$, and have 0 in the rest of the positions. For simplicity, we write a length- $(n + 2)$ sequence $s_1 s_2 \dots s_{n+2}$ as $s_1 s_2 s_{i+2} s_{j+2} \oplus_{ij} s_3 s_4 \dots s_{i+1} s_{i+3} \dots s_{j+1} s_{j+3} \dots s_{n+2}$. Formally, the leaf labels in C_e , from left to right, are

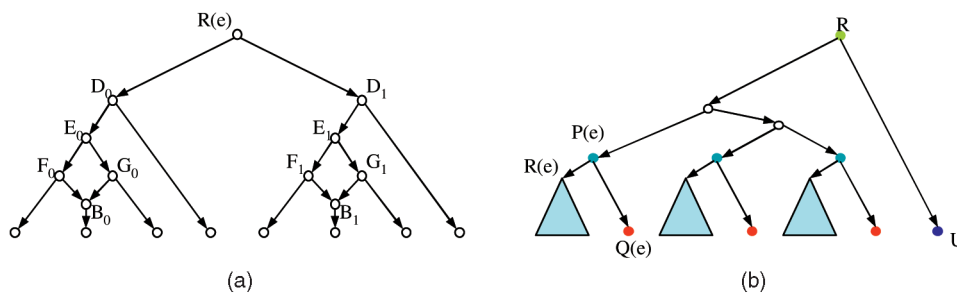


Fig. 3. (a) An edge network C_e . (b) A network \mathcal{N} corresponding with a graph of three edges.

$0010 \oplus_{ij} 0^{n-4}$, $0000 \oplus_{ij} 0^{n-4}$, $1101 \oplus_{ij} 0^{n-4}$, $1111 \oplus_{ij} 0^{n-4}$,
 $0001 \oplus_{ij} 0^{n-4}$, $0011 \oplus_{ij} 0^{n-4}$, $1110 \oplus_{ij} 0^{n-4}$, $1100 \oplus_{ij} 0^{n-4}$,

and the label of $Q(e)$ is 1^{n+2} . Finally, we label U by 110^n .

To complete the reduction, we will prove that the internal nodes can be labeled and the recombination nodes can be specified in a way such that: 1) The score of the resulting network is minimal and 2) for each $e = (i, j)$, the label of $R(e)$ is either $1110 \oplus_{ij} 0^{n-4}$ or $1101 \oplus_{ij} 0^{n-4}$. We start with the following fact:

Lemma 3.1. *Let $e = (u_i, u_j) \in E(G)$. The edge subnetwork C_e has parsimony score 10 if it is labeled in one of the following two ways:*

1. Label $R(e)$, D_0 , E_0 , D_1 , E_1 , and G_1 by $1110 \oplus_{ij} 0^{n-4}$, B_0 by $0000 \oplus_{ij} 0^{n-4}$, F_0 by $0010 \oplus_{ij} 0^{n-4}$, G_0 by $1100 \oplus_{ij} 0^{n-4}$, and B_1, F_1 by $0011 \oplus_{ij} 0^{n-4}$. In addition, we set the crossover position at B_0 to 3 with (F_0, B_0) as the p -edge, the crossover position at B_1 to $j+3$ with (F_1, B_1) as the p -edge.
2. Label $R(e)$, D_0 , E_0 , G_0 , D_1 , E_1 by $1101 \oplus_{ij} 0^{n-4}$, B_1 by $0011 \oplus_{ij} 0^{n-4}$, F_1 by $0001 \oplus_{ij} 0^{n-4}$, G_1 by $1111 \oplus_{ij} 0^{n-4}$, and B_0, F_0 by $0000 \oplus_{ij} 0^{n-4}$. In addition, we set the crossover position at the recombination node B_0 to $j+3$ with (F_0, B_0) as the p -edge, the crossover position at the recombination node B_1 to 3 with (F_1, B_1) as the p -edge.

Proof.

1. Under the labeling, the score is 3 on E_1F_1 , 2 on E_0F_0 , 1 on E_0G_0 , the right edges below D_0, G_0, D_1 , and the left edge below F_1 . The score on other edges is 0.
2. It holds by symmetry. \square

In fact, the edge subnetwork under the labelings in the above lemma has the minimum score as shown below.

Lemma 3.2. *Let $e = (u_i, u_j)$ and l be a labeling of the edge network C_e (with the root $R(e)$) in which a node u is labeled with $l(u)$. Then, the edge network C_e under l has score*

$$s(C_e, l) \geq 10 + \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k].$$

Furthermore, if the equality holds, then

$$l(R(e))[i+2] \neq l(R(e))[j+2].$$

Proof. Let $X = \{1, 2, i+2, j+2\}$ and $Y = \{1, 2, \dots, n\}$. We have:

$$s_p(C_e, l) = \sum_{k \in X} s_p(T_k) + \sum_{k \in Y \setminus X} s_p(T_k),$$

where T_k is the binary subtree of C_e specified by the crossover positions at the recombination nodes B_0 and B_1 in which each node u is labeled with $l(u)[k]$, the k th character of $l(x)$.

For any position, $k \in Y \setminus X$, all of the leaves of T_k are labeled with 0. Hence, its score $s_p(T_k)$ is at least 1 if $l(R(e))[k] = 1$. So, $\sum_{k \in Y \setminus X} s_p(T_k) \geq \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k]$. Therefore, we need only prove that $\sum_{k \in X} s_p(T_k) \geq 10$. To

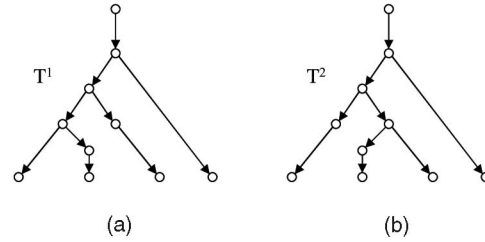


Fig. 4. The binary trees T^1 and T^2 in the proof of Lemma 3.2.

this end, for each $k \in X$, we let T'_k and T''_k denote the subtrees of T_k rooted at D_0 and D_1 , respectively. Clearly,

$$s_p(T_k) \geq s_p(T'_k) + s_p(T''_k).$$

First, T'_k is identical to either T^1 or T^2 in Fig. 4. By Fitch's algorithm, we have

1. If T'_1 and T'_2 are T^1 , then their best scores $s_p(T'_1)$ and $s_p(T'_2)$ are 1; if they are T^2 , their scores are at least 2.
2. If T'_{i+2} is T^1 , its score is at least 2; if it is T^2 , its best score is 1.
3. If T'_{j+2} is T^1 , its best score is 1; if it is T^2 , its score is at least 2.

We have $1 < 2 < i+2 < j+2$ by assumption on $e = (u_i, u_j)$. Hence, no matter how we set the crossover position for B_0 , it is not possible that T'_1, T'_2 , and T'_{j+2} are T^1 , but T'_{i+2} is T^2 . This implies that $\sum_{k \in X} s_p(T'_k) \geq 5$. Furthermore, the equality holds only if T'_1 and T'_2 are T^1 , and T'_{i+2} is identical to T'_{j+2} .

Similarly, the same conclusion holds for T''_1, T''_2, T''_{i+2} , and T''_{j+2} . Thus, $\sum_{k \in X} s_p(T_k) \geq 10$. For the equality to hold, $\sum_{k \in X} s_p(T'_k) = 5$ and $\sum_{k \in X} s_p(T''_k) = 5$. This implies that T_1 and T_2 are identical and T_{i+2} and T_{j+2} have the same topology. In addition, by the Fitch algorithm, another necessary condition for the equality to hold is $l(R(e))[i+2] \neq l(R(e))[j+2]$. This concludes the proof. \square

Lemma 3.3. *There is a labeling l of the network \mathcal{N} such that it achieves the minimum score of \mathcal{N} and $l(R(e)) = 1110 \oplus_{ij} 0^{n-4}$ or $1101 \oplus_{ij} 0^{n-4}$ for any edge e .*

Proof. Let l be a labeling of \mathcal{N} that gives the minimum score. Assume that $l(R(e)) \neq 1110 \oplus_{ij} 0^{n-4}, 1101 \oplus_{ij} 0^{n-4}$ for some $e = (u_i, u_j)$. We obtain a new labeling, l' , from l such that $s_p(\mathcal{N}, l') \leq s_p(\mathcal{N}, l)$ as follows:

- Let BT be the tree obtained from \mathcal{N} by removing all edge clusters. Notice that BT includes the edge $(P(e), Q(e))$ for each e . For each node u of BT , let $l'(u) = 11 \cdot l(u)[3, n+2]$, i.e., we replace the first two sites of $l(u)$ by 11.
- For each edge subnetwork C_e , if $l(R(e))[i+2] = 1$, then label $R(e)$ by $1110 \oplus_{ij} 0^{n-4}$ and label other nodes below $R(e)$ according to Lemma 3.1.1; if $l(R(e))[i+2] = 0$, label $R(e)$ by $1101 \oplus_{ij} 0^{n-4}$ and label other nodes below $R(e)$ according to Lemma 3.1.2.

By definition,

$$s_p(\mathcal{N}, l) = s_p(BT, l) + \sum_{e \in E(G)} [s_p(C_e, l) + h(l(P(e)), l(R(e)))]$$

and

$$s_p(\mathcal{N}, l') = s_p(BT, l') + \sum_{e \in E(G)} [s_p(C_e, l') + h(l'(P(e)), l'(R(e)))].$$

Since, for each edge $(u, v) \in BT$,

$$\begin{aligned} l'(u)[1, 2] &= l'(v)[1, 2] = 11, \\ l'(u)[3, n+2] &= l(u)[3, n+2], \end{aligned}$$

and

$$l'(v)[3, n+2] = l(v)[3, n+2],$$

we have $h(l'(u), l'(v)) \leq h(l(u), l(v))$ for every edge $(u, v) \in E(BT)$. Hence,

$$\begin{aligned} s_p(BT, l') &= \sum_{(u,v) \in E(BT)} h(l'(u), l'(v)) \leq \sum_{(u,v) \in E(BT)} h(l(u), l(v)) \\ &= s_p(BT, l). \end{aligned}$$

Therefore, we only need to prove that

$$\begin{aligned} s_p(C_e, l') + h(l'(P(e)), l'(R(e))) \\ \leq s_p(C_e, l) + h(l(P(e)), l(R(e))) \end{aligned}$$

by considering the following cases:

Case 1. $l(R(e))[i+2] \neq l(R(e))[j+2]$. Without loss of generality, we assume $l(R(e))[i+2] = 1$. Then, $l'(R(e)) = 1110 \oplus_{ij} 0^{n-4}$ and

$$h(l(R(e))[3, n+2], l'(R(e))[3, n+2]) = \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k].$$

By Lemma 3.2, $s_p(C_e, l) \geq 10 + \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k]$. On the other hand, by Lemma 3.1, $s_p(C_e, l') = 10$. But, $h(l'(P(e)), l'(R(e)))$ increases. Since $l'(P(e))[1, 2] = l'(R(e))[1, 2]$ and $l'(P(e))[3, n+2] = l(P(e))[3, n+2]$, the increment $h(l'(P(e)), l'(R(e))) - h(l(P(e)), l(R(e)))$ is at most

$$h(l(R(e))[3, n+2], l'(R(e))[3, n+2]) = \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k].$$

Therefore, $s_p(\mathcal{N}, l') \leq s_p(\mathcal{N}, l)$.

Case 2. $l(R(e))[i+2] = l(R(e))[j+2]$. In this case,

$$\begin{aligned} h(l(R(e))[3, n+2], l'(R(e))[3, n+2]) \\ = \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k] + 1. \end{aligned}$$

By Lemma 3.2, $s_p(C_e, l) > 10 + \sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k]$. Similarly to Case 1, l' reduces the score of the subnetwork C_e by $\sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k] + 1$ and increases the score of the edge $P(e)R(e)$ at most

$$\sum_{k \neq 1, 2, i+2, j+2} l(R(e))[k] + 1.$$

Hence, we also have $s_p(\mathcal{N}, l') \leq s_p(\mathcal{N}, l)$.

This concludes the proof. \square

Theorem 3.2. *The small parsimony problem is NP-hard for the recombination galled networks satisfying one of the following two conditions:*

1. Both the recombination edges and crossover position are not specified for each recombination node.
2. The p and s -edges are specified, but the crossover position is not known for each recombination node.

Proof. We first prove the NP-hardness for Condition 1, then show how to modify the proof for Condition 2. The theorem follows from the fact that G has a vertex cover of size t if and only if $s_p(\mathcal{N}) \leq (n+9)|E(G)| + t$, where $|E(G)|$ denotes the number of edges in G .

Let V' be a vertex cover of G such that $|V'| = t$. For each edge $e = (u_i, u_j)$ of G , u_i or u_j is in V' . We select only one of u_i and u_j that is in V' even if both are in V' . Without loss of generality, we assume that $u_i \in V'$ is selected. We label the internal nodes in C_e according to Lemma 3.1.1. Furthermore, we label all the other nodes (including $P(e)$ for each edge) by the sequence $11 \cdot x_1 x_2 \cdots x_n$, where $x_k \in \{0, 1\}$ and $x_k = 1$ if and only if $u_k \in V'$. Under the resulting labeling l , the score is $|V'| = t$ on the edge (R, U) , $t-1$ on the edge $(P(e), R(e))$, $n-t$ on the edge $(P(e), Q(e))$, and 0 on the other edges in the top part of the network \mathcal{N} . Using Lemma 3.1, we obtain

$$\begin{aligned} s_p(\mathcal{N}) \leq s_p(\mathcal{N}, l) &= |E(G)|[10 + (t-1) + (n-t)] + t \\ &= |E(G)|(n+9) + t. \end{aligned}$$

Conversely, assume that $s_p(\mathcal{N}) \leq |E(G)|(n+9) + t$. By Lemma 3.3, there is a labeling l of \mathcal{N} having the minimum score in which the node $R(e)$ in the edge network corresponding to $e = (u_i, u_j)$ is labeled with either $1110 \oplus_{ij} 0^{n-4}$ or $1101 \oplus_{ij} 0^{n-4}$. Let $V' = \{u_k | \exists e \in E(G) \text{ s.t. } l(R(e)) = 11 \cdot 0^{k-1} 10^{n-k}\}$. Clearly, V' is a vertex cover of G .

Let T be the tree obtained by removing from \mathcal{N} all of the edge networks below $R(e)$ (not including $R(e)$ and $Q(e)$ for each edge e). Applying the Fitch algorithm for the small parsimony problem on T , we can easily show that $s_p(T, l) \geq (n-1)|E(G)| + |V'|$. Hence,

$$\begin{aligned} s_p(\mathcal{N}, l) \\ \geq s_p(T, l) + 10|E(G)| \\ \geq (n-1)|E(G)| + |V'| + 10|E(G)| \\ = (n+9)|E(G)| + |V'|. \end{aligned}$$

Since $s_p(\mathcal{N}) = s_p(\mathcal{N}, l) \leq |E(G)|(n+9) + t$, we have that $|V'| \leq t$, i.e., G has a vertex cover of size t . This concludes the proof for Condition 1.

Now, we show how to modify the proof for Condition 2. Note that, in Lemma 3.1, the p -edge of B_0 is always set to $F_0 B_0$; the p -edge of B_1 is always set to $F_1 B_1$. Hence, when we assign the p -edge of B_0 and B_1 to $F_0 B_0$ and $F_1 B_1$, respectively, in the reconstructed network for Condition 1, all of the discussion above remains the

same. This shows that the NP-hardness is also true for Condition 2. \square

3.3 Remarks on General Phylogenetic Networks

In [18], Nakhleh et al. defined the *block parsimony score* for studying phylogenetic networks. Given a phylogenetic network \mathcal{N} with leaves labeled by length- d sequences, let $\mathcal{T}(\mathcal{N})$ be the set of all possible phylogenetic subtrees that have the same leaves as \mathcal{N} . Furthermore, let $\mathcal{N}_{i,j}$ be the network having the same topology with \mathcal{N} , but each leaf u is labeled by the subsequence $s[i, j]$ ($1 \leq i < j \leq d$), where s is the label of u in \mathcal{N} and $s[i, j]$ is the substring that contains the letters from position i to position j . We partition each labeling sequence s into k blocks

$$s[1, a_1 - 1], s[a_1, a_2 - 1], \dots, s[a_{k-1}, d],$$

where $1 < a_1 < a_2 < \dots < a_{k-1} < d + 1$. Then, the block parsimony score of \mathcal{N} regarding the partition is

$$bs_p(\mathcal{N}) = \sum_{i=1}^k \min_{T \in \mathcal{T}(\mathcal{N}_{a_i, a_{i+1}-1})} s_p(T).$$

Note that the parsimony score defined by Hein in [8] and [9] is a special case of the block parsimony score where each block contains only one site.

Since the network constructed in the proof of Theorem 3.1 has leaves labeled with a character, the specification of crossover positions there has the same effect as the specification of block partitions for the same network. Therefore, it holds that the small parsimony problem is NP-hard for the phylogenetic networks regarding to the block parsimony score. This answers the open question in [18]. Since the set cover problem cannot be approximated with ratio $c(\log n)$ for some constant c , the reduction constructed there also indicates that approximating the block parsimony score of a network \mathcal{N} within a ratio of $c(\log n)$ is NP-hard.

The small parsimony problem remains NP-hard for the galled phylogenetic networks. We can prove this fact using a reduction from the vertex cover problem almost identical to the one in the proof of Theorem 3.2 except for labeling leaves. Here, each leaf in the galled network constructed from a graph is labeled with the length- n suffix of the corresponding label sequence used there and there is only one block. The proof follows from the fact that the graph G has a vertex cover of size t if and only if the block parsimony score of \mathcal{N} is $(n + 5)|E(G)| + t$ regarding some block partition.

4 A DYNAMIC PROGRAMMING APPROACH

We have known that the small parsimony problem is polynomial-time computable for the recombination networks in which the recombination edges and crossover position are fully specified for each recombination node (Theorem 2.1). We have also proved that if the recombination nodes are not completely specified, the small parsimony problem becomes NP-hard even for the galled networks. In this section, we present a dynamic programming algorithm for solving the small parsimony problem. The algorithm is linear in sequence length and the number

of sequences, but exponential in the number of recombination nodes.

We first consider the recombination networks in which, for each recombination node, the p -edge and s -edge are specified, but the crossover position is not set. Let \mathcal{N} be such a recombination network with h recombination nodes, $n - h$ tree nodes, and leaves labeled by length- d sequences. Assume the h recombination nodes are r_1, r_2, \dots, r_h with crossover positions c_1, c_2, \dots, c_h , respectively. Since each c_i takes d possible values, exhaustive search takes $O(dnd^h)$ time for solving the small parsimony problem on \mathcal{N} . However, the dynamic programming proposed in the rest of this section just takes $O(dn2^{2h})$ time instead.

For each column i ($1 \leq i \leq d$) of the sequences, the specific values of the crossover positions c_1, c_2, \dots, c_h define a unique tree T_i , which contains the p -edge of r_i if $i < c_i$ and the s -edge of r_i otherwise. It is easy to see that the defined subtree series $T_1, T_2, T_3, \dots, T_d$ satisfies the following property:

Monotone property. For a recombination node r_j , if T_i contains the s -edge of r_j , then T_k must contain the s -edge of r_j for any $i < k \leq d$.

Note that there are 2^h subtrees in \mathcal{N} that are obtained by removing exactly one recombination edge of each recombination node. Let $Trees_{\mathcal{N}} = \{G_1, G_2, \dots, G_{2^h}\}$ be the set of such subtrees. A series of d subtrees $(G_{i_1}, G_{i_2}, \dots, G_{i_d})$ from $Trees_{\mathcal{N}}$ is said to be *compatible* if it satisfies the monotone property for every recombination node in \mathcal{N} . Obviously, a compatible series of d subtrees $(T'_1, T'_2, \dots, T'_d)$ defines a unique crossover position

$$c_i = \min\{j \mid T'_j \text{ contains the } s\text{-edges of } r_i\}$$

for every recombination node r_i , $i \leq h$. Hence, there is a one-to-one correspondence between the set $\Gamma_{\mathcal{N}}$ of compatible series of d subtrees from $Trees_{\mathcal{N}}$ and the ways of setting the crossover positions of the h recombination nodes. Therefore,

$$s_p(\mathcal{N}) = \min_{(T_i)_{i=1}^d \in \Gamma_{\mathcal{N}}} \sum s_p(T_i).$$

Now, we consider $Trees_{\mathcal{N}}$ as a set of trees in which the elements are ordered. Let $T', T'' \in Trees_{\mathcal{N}}$. We denote $T' \ll T''$ if, for each recombination node r_i , the fact that T' contains the s -edge of r_i implies that T'' also contains the s -edge of r_i . Obviously, we have the following fact:

Lemma 4.1.

1. For any $T \in Trees_{\mathcal{N}}$, $T \ll T$.
2. For any $T''_1, T''_2, T''_3 \in Trees_{\mathcal{N}}$, if $T''_1 \ll T''_2$ and $T''_2 \ll T''_3$, then $T''_1 \ll T''_3$.
3. Let $1 \leq k \leq d$. Assume that $T''_i \in Trees_{\mathcal{N}}$, $1 \leq i \leq k$. Then, $(T''_1, T''_2, \dots, T''_k)$ is compatible if and only if $T''_1 \ll T''_2 \ll \dots \ll T''_k$.

For any $1 \leq j \leq d$ and $1 \leq k \leq 2^h$, let $\Phi(j, k)$ be the set of compatible series of j subtrees in which the last subtree is G_k . Since (G_k, G_k, \dots, G_k) is a compatible sequence, $\Phi(j, k)$ is not empty for any j . Set

$$V(j, k) = \min_{(T'_1, T'_2, \dots, T'_j) \in \Phi(j, k)} \sum_{i=1}^j s_p(T'_i).$$

Here, $V(j, k)$ can be considered as a parsimony score of the network \mathcal{N} on the first j sites. Then, it is not hard to see

$$s_p(\mathcal{N}) = \min_{1 \leq k \leq 2^h} V(d, k)$$

and, furthermore, we have:

Lemma 4.2. *The following recurrence relation holds on j :*

$$V(j, k) = \min_{G_{k'}: G_{k'} \ll G_k} (V(j-1, k') + s_p(G_k)).$$

Proof. Assume $(T''_1, T''_2, \dots, T''_{j-1}, T''_j = G_k) \in \Phi(j, k)$ such that

$$V(j, k) = \min_{(T'_1, T'_2, \dots, T'_j) \in \Phi(j, k)} \sum_{i=1}^j s_p(T'_i) = \sum_{i=1}^j s_p(T''_i).$$

Let $T''_{j-1} = G_{k'}$. Assume $\sum_{i=1}^{j-1} s_p(T''_i) > V(j-1, k')$. Then, there are $F''_1, F''_2, \dots, F''_{j-1} = G_{k'}$ such that $F''_1 \ll F''_2 \ll \dots \ll F''_{j-1}$ and $V(j-1, k') = \sum_{i=1}^{j-1} s_p(F''_i)$. By Lemma 4.1, $F''_1 \ll F''_2 \ll \dots \ll F''_{j-1} \ll T''_j$ and, in addition, $\sum_{i=1}^{j-1} s_p(F''_i) + T''_j < \sum_{i=1}^j s_p(T''_i) = V(j, k)$, a contradiction to the definition of $V(j, k)$. Therefore, if

$$(T''_1, T''_2, \dots, T''_{j-1} = G_{k'}, T''_j = G_k) \in \Phi(j, k)$$

such that $\sum_{i=1}^j s_p(T''_i) = V(j, k)$, then

$$\sum_{i=1}^{j-1} s_p(T''_i) = V(j-1, k').$$

This implies that

$$\begin{aligned} V(j, k) &= \min_{G_{k'}: G_{k'} \ll G_k} \left(\min_{(T'_1, T'_2, \dots, T'_{j-1}) \in \Phi(j, k')} \sum_{i=1}^{j-1} s_p(T'_i) + s_p(G_k) \right) \\ &= \min_{G_{k'}: G_{k'} \ll G_k} (V(j-1, k') + s_p(G_k)). \end{aligned}$$

This proves the recurrence relation. \square

Therefore, the tabular computation of all of the $V(j, k)$, $1 \leq j \leq d$ and $1 \leq k \leq 2^h$, takes $O(dn2^{2h})$ by calling the Fitch algorithm since each subtree has at most n nodes and the above recurrence relation contains as many as 2^h terms. This proves that the small parsimony problem can be solved in time $O(dn2^{2h})$.

In general, when all of the s -edges and p -edges are not specified for the recombination nodes, we may simply examine all of the possible 2^h assignments of the s -edges and p -edges for the recombination nodes and run the above dynamic algorithm for each assignment. Therefore, we have the following theorem:

Theorem 4.1. *The small parsimony problem can be solved by a dynamic programming approach in time $O(dn2^{3h})$.*

In practice, the sequence length is quite larger than 8 and, so, our proposed method is much more efficient than the straightforward method.

5 CONCLUSION

In this paper, we have customized the parsimony score for recombination networks and proved the small parsimony problem to be NP-hard even for galled recombination networks. The NP-hardness proof can be generalized to the reticulation networks. We also present a dynamic programming algorithm for solving the small parsimony problem, whose time complexity is linear in sequence length and the number of sequences but exponential in the number of recombination nodes. Although the proposed algorithm is more efficient than the naive method, it is still not fast enough for practical reconstruction purposes. As a future research topic, we will continue to study how to efficiently reconstruct a recombination network model from sequence data.

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for helpful suggestions for revising the manuscript. L. Zhang and C. Thach Nguyen gratefully acknowledge the ARF grant 146-000-068-112 for partially supporting this project.

REFERENCES

- [1] G. Bourque and L.X. Zhang, "Models and Methods in Comparative Genomics," *Advances in Computer Science*, vol. 68, pp. 59-104, 2006.
- [2] C. Choy, J. Jansson, K. Sadakane, and W.-K. Sung, "Computing the Maximum Agreement of Phylogenetic Networks," *Theoretical Computer Science*, vol. 335, pp. 93-107, 2005.
- [3] W.F. Doolittle, "Phylogenetic Classification and the Universal Tree," *Science*, vol. 284, pp. 2124-2129, 1999.
- [4] W.M. Fitch, "Toward Defining the Course of Evolution: Minimum Change for Specific Tree Topology," *J. Zoological System*, vol. 20, pp. 406-416, 1971.
- [5] D. Gusfield and V. Bansal, "A Fundamental Decomposition Theory for Phylogenetic Networks and Incompatible Characters," *Proc. RECOMB '05*, pp. 217-232, 2005.
- [6] D. Gusfield, S. Eddhu, and C. Langley, "Optimal, Efficient Reconstruction of Phylogenetic Networks with Constrained Recombination," *J. Bioinformatics and Computational Biology*, vol. 2, no. 1, pp. 173-213, 2004.
- [7] D. Gusfield, S. Eddhu, and C. Langley, "The Fine Structure of Galls in Phylogenetic Networks," *Inform. J. Computing*, vol. 16, no. 4, pp. 459-469, 2004.
- [8] J. Hein, "Reconstructing the History of Sequences Subject to Gene Conversion and Recombination," *Math. Bioscience*, vol. 98, pp. 185-200, 1990.
- [9] J. Hein, "A Heuristic Method to Reconstruct the History of Sequences Subject to Recombination," *J. Molecular Evolution*, vol. 20, pp. 402-411, 1993.
- [10] J. Huang, N. Mullapudi, T. Sicheritz-Ponten, and J.C. Kissinger, "A First Glimpse into the Pattern and Scale of Gene Transfer in Apicomplexa," *Int'l J. Parasitology*, vol. 34, no. 3, pp. 265-74, 2004.
- [11] D.H. Huson et al., "Reconstruction of Reticulate Networks from Gene Trees," *Lecture Notes in Computer Science* vol. 3500, pp. 233-249, 2005.
- [12] D.H. Huson, T. Dezulian, T. Klopper, and M. Steel, "Phylogenetic Super-Networks from Partial Trees," *IEEE Trans. Computational Biology and Bioinformatics*, vol. 1, pp. 151-158, 2004.
- [13] D.H. Huson and T.H. Klopper, "Computing Recombination Networks from Binary Sequences," *Bioinformatics*, vol. 21, supplement ii, pp. ii159-ii165, 2005.
- [14] T.N.D. Huynh, J. Jansson, N.B. Nguyen, and W.-K. Sung, "Constructing a Smallest Refining Galled Phylogenetic Network," *Lecture Notes in Computer Science*, vol. 3500, pp. 265-280, 2005.
- [15] J. Jansson, N.B. Nguyen, and W.-K. Sung, "Algorithms for Combining Rooted Triplets into a Galled Phylogenetic Network," *Proc. ACM-SIAM Symp. Discrete Algorithms (SODA '05)*, pp. 349-358, 2005.

- [16] J.G. Lawrence and H. Ochman, "Reconciling the Many Faces of Lateral Gene Transfer," *Trends Microbiology*, vol. 10, pp. 1-4, 2002.
- [17] D.A. Morrison, "Networks in Phylogenetic Analysis: New Tools for Population Biology," *Int'l J. Parasitology*, vol. 35, no. 5, pp. 567-82, 2005.
- [18] L. Nakhleh et al., "Reconstructing Phylogenetic Networks Using Maximum Parsimony," *Proc. 2005 IEEE Computational Systems Bioinformatics Conf.*, pp. 93-102, 2005.
- [19] L. Nakhleh et al., "Reconstructing Reticulate Evolution in Species—Theory and Practice," *J. Computational Biology*, vol. 12, pp. 796-811, 2005.
- [20] K.E. Nelson et al., "Evidence for Lateral Gene Transfer between Archaea and Bacteria from Genome Sequence of *Thermotoga Maritima*," *Nature*, vol. 27, pp. 323-329, 1999.
- [21] M. Nordborg and S. Tavaré, "Linkage Disequilibrium: What History Has to Tell Us," *Trends in Genetics*, vol. 18, pp. 83-90, 2002.
- [22] L.S. Wang, K.Z. Zhang, and L.X. Zhang, "Perfect Phylogenetic Networks with Recombination," *J. Computational Biology*, vol. 8, no. 1, pp. 69-78, 2001.



C. Thach Nguyen received the Bachelor of Computing degree from the School of Computing at the National University of Singapore in 2005. He is currently a graduate student in the Department of Computer Science and Engineering at the University of Washington, Seattle. His research interests include algorithms, theory, and computational biology.



Nguyen Bao Nguyen received the Bachelor of Computing degree from the School of Computing at the National University of Singapore in 2006. She is currently a graduate student in the Department of Computer Science and Engineering at the University of Washington, Seattle. Her research interests are mainly in the fields of computational biology and algorithms.



Wing-Kin Sung received both the BSc and PhD degrees from the Department of Computer Science at the University of Hong Kong. Currently, he is an assistant professor in the Department of Computer Science at the National University of Singapore, Singapore. He also works as a senior group leader in the Department of Information and Mathematical Science, Genome Institute of Singapore, Singapore.



Louxin Zhang received the BS and MS degrees in mathematics from Lanzhou University, China, and the PhD degree in computer science from the University of Waterloo, Canada. He is an associate professor of applied mathematics at the National University of Singapore. His research areas have been in computational biology, bioinformatics, applied combinatorics, and theory of computing. In particular, the focus of his current research is on sequence comparison, gene duplication in comparative genomics. He has coauthored more than 40 scholarly research articles. He serves on the editorial boards of the *International Journal of Bioinformatics Research and Applications* and *Genomics, Proteomics & Bioinformatics* and has also served on the program committees of various international conferences and workshops.

► For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.