JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 12, Number 6, 2005

Pp. 796–811

Reconstructing Reticulate Evolution in Species—Theory and Practice

LUAY NAKHLEH,[1] TANDY WARNOW,[2] C. RANDAL LINDER,[3] and KATHERINE ST. JOHN[4]

ABSTRACT

We present new methods for reconstructing reticulate evolution of species due to events such as horizontal transfer or hybrid speciation; both methods are based upon extensions of Wayne Maddison's approach in his seminal 1997 paper. Our first method is a polynomial time algorithm for constructing phylogenetic networks from two gene trees contained inside the network. We allow the network to have an arbitrary number of reticulations, but we limit the reticulation in the network so that the cycles in the network are node-disjoint ("galled"). Our second method is a polynomial time algorithm for constructing networks with one reticulation, where we allow for errors in the estimated gene trees. Using simulations, we demonstrate improved performance of this method over both NeighborNet and Maddison's method.

Key words: phylogenetic networks, reticulate evolution, hybrid speciation.

1. INTRODUCTION

THE MOTIVATION FOR THIS PAPER IS THE PROBLEM of reconstructing accurate evolutionary history in the presence of reticulation events, such as hybrid speciation (where organisms hybridize and create new species) or horizontal transfer (via hybridization or viral transmission, for example). Both types of reticulation events are sufficiently common to be of serious concern to systematists; hybrid speciation is common in some very large groups of organisms: plants, fish, amphibians, and many lineages of invertebrates, and horizontal gene transfer appears to be very common in bacteria (Lawrence and Ochman, 2002) with lower levels being evident in many multicellular groups. Such evolutionary histories cannot be adequately represented using trees; instead, phylogenetic networks (which are basically directed acyclic graphs, coupled with time constraints) are used.

Several methods are currently used to build phylogenetic networks from gene datasets, although not all methods were designed for this purpose; of these, NeighborNet by Bryant and Moulton (2002) and a

[1]Department of Computer Science, Rice University, Houston, Texas 77005.
[2]Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712.
[3]Section of Integrative Biology, School of Biological Sciences, University of Texas at Austin, Austin, Texas 78712.
[4]Department of Mathematics and Computer Science, Lehman College, City University of New York, Bronx, New York 10468.

method by Wayne Maddison (1997) are the most relevant to this paper. NeighborNet uses a "combined analysis" approach because it combines sequence datasets by concatenation and then seeks the phylogenetic network on the basis of the distance matrix produced by the combined dataset. Maddison, on the other hand, uses a separate analysis where the network can be reconstructed by first inferring individual gene trees from separate sequence datasets and then reconciling the trees into a network.

In this paper, we consider the inference of "gt-networks" (for "galled tree" networks, which is the terminology used by Gusfield *et al.* [2003]); these are phylogenetic networks in which reticulation events are constrained so as to be evolutionarily independent of each other (see Gusfield *et al.* [2003] for a biological justification of the model). This model was first introduced by Wang *et al.* (2001), and later formalized and further pursued by Gusfield *et al.* (2003).

We present polynomial time algorithms that provably reconstruct accurate phylogenetic networks, provided that accurate gene trees can be obtained. We also present polynomial time algorithms for reconstructing phylogenetic networks from inaccurate gene trees, and we demonstrate the improvement in accuracy of these methods over two previous methods for phylogenetic network reconstruction in simulation.

The rest of the paper is organized as follows. In Section 2, we briefly describe phylogenetic networks, including the definition of gt-networks. In Section 3, we briefly describe two of the evolutionary events that necessitate the use of phylogenetic networks; we also review Maddison's approach and discuss its limitations. In Section 4, we present our efficient algorithm for reconciling accurate gene trees into a gt-network. In Section 5, we present a linear time algorithm for the following combinatorial problem: given two trees t_1 and t_2, does there exist a pair of trees T_1 and T_2 refining t_1 and t_2, respectively, such that T_1 and T_2 are the two induced trees in a gt-network with one reticulation? We show how to use this algorithm for reconstructing phylogenetic networks in practice in Section 6. In Section 7, we summarize the results of a simulation study comparing the performance of this method (which we call SPNET, for "Species Network") to NeighborNet. We close in Section 8 with final remarks and directions for future research.

2. NETWORKS AND GT-NETWORKS

2.1. Background

2.1.1. Graph-theoretic definitions. Given a (directed) graph G, $E(G)$ denotes the set of (directed) edges of G, and $V(G)$ denotes the set of nodes of G. We write (u, v) to denote a directed edge from node u to node v, in which case u is the *tail*, v the *head* of the edge, and u is a *parent* of v. The *indegree* of a node v is the number of edges whose head is v, while the *outdegree* of v is the number of edges whose tail is v. A directed path of length k from u to v in G is a sequence $u_0u_1 \cdots u_k$ of nodes with $u = u_0$, $v = u_k$, and $\forall i,\ 1 \leq i \leq k,\ (u_{i-1}, u_i) \in E(G)$; we say that u is the tail of p and v is the head of p.

Node v is *reachable* from u in G, denoted $u \rightsquigarrow v$, if there is a directed path in G from u to v; we then also say that u is an *ancestor* of v. Given a tree T and a subset L' of the leaves, we write $T|_{L'}$ to denote the subtree obtained by restricting T to leaves L', i.e., by removing all leaves not in L' and all incident edges. If X is a subtree of T, we denote by $T \setminus X$ the tree obtained by removing subtree X from T.

We denote by $L(T)$ the leaf set of a tree T. If T is not binary, we call it an *unresolved* tree. An undirected path p of length k between u and v in a rooted tree T is a sequence $u_0u_1 \cdots u_k$ of nodes with $u = u_0$, $v = u_k$, and $\forall i,\ 1 \leq i \leq k$, either (u_{i-1}, u_i) or (u_i, u_{i-1}) is an edge of T. If p is an undirected path in tree T whose two endpoints are u and v, we denote by $END(p) = (U, V)$ the two subtrees U and V attached to u and v, respectively, that do not contain any edges from p. We use p to denote the path itself, as well as the edges of the path.

2.1.2. Strict consensus and compatibility trees. Let T be a tree leaf-labeled by a set S of taxa. Each edge e in T induces a *bipartition* $\pi(e) = \{A(e)|B(e)\}$ on the set S, where $A(e)$ is the set of taxa "below" e and $B(e)$ is the set containing the rest of the taxa. We denote by $C(T)$ the set of all bipartitions induced by tree T.

We say that e_1 and e_2 (and their associated bipartitions) are *compatible*, denoted $e_1 \equiv e_2$, if there exists a tree T that induces both $\pi(e_1)$ and $\pi(e_2)$. A set C of bipartitions is compatible if it is pairwise

compatible; i.e., every two bipartitions $c_1, c_2 \in C$ are compatible. Two trees T_1 and T_2 are compatible if the set $C(T_1) \cup C(T_2)$ of bipartitions is compatible. Two trees that are not compatible are called *incongruent trees*.

If we contract an edge in T, thus identifying the endpoints of that edge, we obtain another tree T' on the same leaf set; T is then said to *refine* T', and T' is said to be a *contraction* of T. If T'' is the result of contracting a set of edges in T, then too T'' is a contraction of T, and T is a refinement of T''.

A set of trees is compatible if the trees have a common refinement; its minimal common refinement (called the *compatibility tree*) is unique. For any set of trees, the maximally resolved common contraction (called the *strict consensus tree*) is also unique. Both the compatibility (Gusfield, 1991; Warnow, 1994) and strict consensus trees (Day, 1985) can be found in $O(kn)$ time, where there are k trees on the same set of n leaves.

Given two trees T_1 and T_2, the set $U(T_1, T_2)$ contains all edges of T_1 that are not compatible with T_2; $U(T_2, T_1)$ is defined similarly. Note then that T_1 and T_2 are compatible if $U(T_1, T_2) = U(T_2, T_1) = \emptyset$.

If a tree T has a node v with indegree and outdegree one, we replace the two edges incident to v by a single edge; this operation on T is called *forced contraction*.

2.2. Phylogenetic networks

A *phylogenetic network* $N = (V, E)$ with a set $L \subseteq V$ of n leaves is a directed acyclic graph in which exactly one node has no incoming edges (the root) and all other nodes have either one incoming edge (tree nodes) or two incoming edges (reticulation nodes). The nodes in L have no outgoing edges. Tree edges are those whose head is a tree node, and network edges are those whose head is a reticulation node.

In this paper, we focus on *binary networks*, i.e., networks in which the outdegree of a reticulation node is 1 and the outdegree of a tree node is 2. Further, all trees are binary; i.e., all nodes (except for the leaves) have outdegree 2.

As discussed by Moret *et al.* (2004), reticulation events impose time constraints on the phylogenetic network, which we now briefly review. A phylogenetic network $N = (V, E)$ defines a partial order on the set V of nodes. Based on this partial order, we assign times to the nodes of N, associating time $t(u)$ with node u. If there is a directed path p from node u to node v, such that p contains at least one tree edge, then we must have $t(u) < t(v)$ (in order to respect the time flow). If $e = (u, v)$ is a network edge, then we must have $t(u) = t(v)$ (because a reticulation event is, at the scale of evolution, an instantaneous process).

Given a network N, we say that p is a *positive-time directed path* from u to v, if p is a directed path from u to v and p contains at least one tree edge. Given a network N, two nodes u and v cannot coexist in time if there exists a sequence $P = \langle p_1, p_2, \ldots, p_k \rangle$ of paths such that (1) p_i is a positive-time directed path, for every $1 \leq i \leq k$, (2) u is the tail of p_1, and v is the head of p_k, and (3) for every $1 \leq i \leq k-1$, there exists a reticulation node whose two parents are the head of p_i and the tail of p_{i+1}.

If two nodes u and v cannot coexist in time, then they cannot be "involved" in a reticulation event. In other words, u and v cannot be the two parents of a hybrid (i.e., there does not exist a reticulation node w such that (u, w) and (v, w) are edges in the network), nor can there be a horizontal gene transfer between them (i.e., neither (u, v) nor (v, u) can be an edge in the network). This property is further discussed by Moret *et al.* (2004), Maddison (1997), and Page and Charleston (1998).

2.3. gt-Networks

In this paper, we assume a biologically motivated restricted class of phylogenetic networks, called *gt-networks*, proposed by Wang *et al.* (2001) and Gusfield *et al.* (2003).

Definition 1. *In a phylogenetic network N, let w be a node that has two directed paths out of it that meet at a reticulation node x. Those two directed paths together define a* reticulation cycle *Q. Node w is called the* coalescent node *of Q, and x is the* reticulation node *of Q.*

Definition 2. *A reticulation cycle in a phylogenetic network that shares no nodes with any other reticulation cycle is called a* gall.

Definition 3. *We denote by Q_x^w a gall whose coalescent node is w and whose reticulation node is x. We denote by $E(Q_x^w)$ the set of all edges on gall Q; formally, $E(Q_x^w) = \{e : e$ is an edge on a directed path from w to $x\}$. The set $RE(Q_x^w)$ (for "reticulation edges") denotes the edges whose head is x, i.e., the edges incident into x.*

When the context is clear, we simply write Q for a gall, without explicitly naming the coalescent and reticulation nodes.

Definition 4. *A phylogenetic network N is called a* gt-network *if every reticulation cycle is a gall.*

Figure 1(a) shows a gt-network N with a gall Q_x^w. The set $E(Q_x^w)$ contains the edges (w, w_1), (w_1, u_1), (w, w_2), (w_2, u_2), (u_1, x), and (u_2, x). The set $RE(Q_x^w)$ contains the two edges (u_1, x) and (u_2, x). Obviously, gt-networks satisfy the synchronization property. In this paper, we assume that there is at least one tree node on each of the two paths from w to x in a gall Q_x^w (otherwise, the network would violate the synchronization property).

We *break* a gall Q_x^w by removing exactly one of the edges in the set $RE(Q_x^w)$.

Definition 5. *A tree T is* induced *by a gt-network N if T can be obtained from N through one of the possible ways of breaking all the galls in N, followed by forced contraction operations on all nodes of indegree and outdegree 1.*

Figures 1(b) and 1(c) show the two possible trees induced by the gt-network N in Fig. 1(a). To obtain the tree in Fig. 1(b), the gall was broken by removing edge (u_1, x) and applying forced contraction to node x; to obtain the tree in Fig. 1(c), the gall was broken by removing edge (u_2, x) and applying forced contraction to node x. In general, given a network N with p reticulation nodes, we say that a tree T is *induced* by N if T can be obtained by removing exactly one of the two edges incoming into each of the p reticulation nodes in N.

Definition 6. *Let Q_x^w be gall in a gt-network N, with $RE(Q) = \{e_1 = (u_1, x), e_2 = (u_2, x)\}$. Further, let w_1 be the parent of u_1, and w_2 be the parent of u_2. Assume tree T_1 is obtained from N by removing edge e_1 and tree T_2 is obtained from N by removing edge e_2. The two directed paths $w \rightsquigarrow w_1$ and $w \rightsquigarrow u_2$ together define a "reticulation path" in T_1, and the two directed paths $w \rightsquigarrow w_2$ and $w \rightsquigarrow u_1$ together define a "reticulation path" in T_2.*

Given a gt-network with m galls, there are 2^m possible ways of breaking the m galls and thus inducing a tree. There is a direct correspondence between the edges and nodes of a gt-network N and a tree T induced by N, and hence we talk about a node or edge of T in N, or a node or edge of N in T (excluding the edges in $RE(Q)$ and the nodes removed by forced contraction).

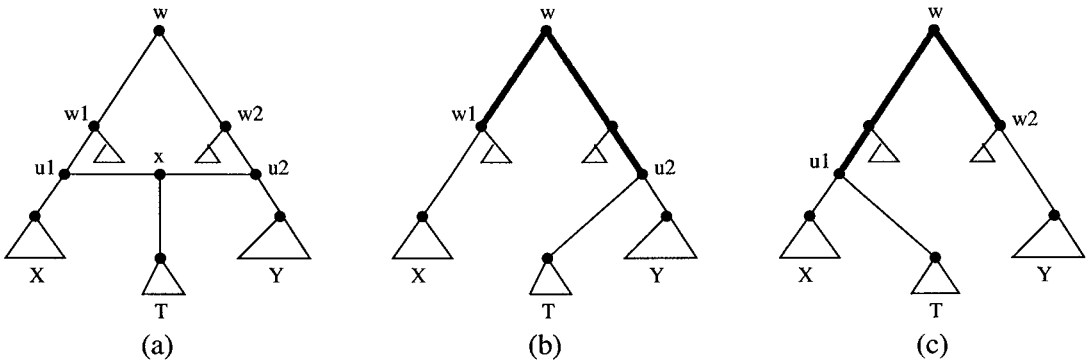

FIG. 1. (**a**) A gall Q whose coalescent and reticulation nodes are w and x, respectively. Panels (**b**) and (**c**) show the two possible ways of "breaking" the gall Q to induce trees T_1 and T_2, respectively. The marked edges in T_1 and T_2 form $RP^Q(T_1)$ and $RP^Q(T_2)$, respectively.

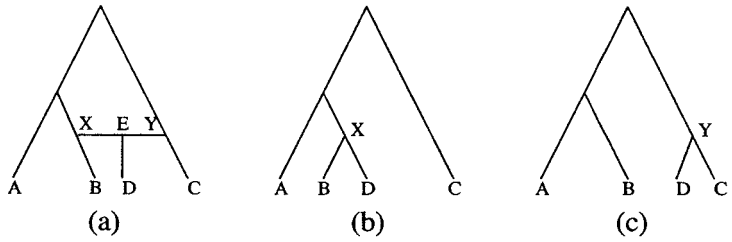

FIG. 2. Hybrid speciation: the species network in (**a**) and its two induced (gene) trees in (**b**) and (**c**).

Definition 7. *We denote by* $RP^Q(T)$ *the* reticulation path *in* T *that results from breaking gall* Q*. The marked edges in tree* T_1 *of Fig. 1(b) form the reticulation path* $RP^Q(T_1)$*, and the marked edges in tree* T_2 *of Fig. 1(c) form the reticulation path* $RP^Q(T_2)$ *(we also use* $RP^Q(T)$ *to denote the edges on the reticulation path in* T*).*

3. RETICULATE EVOLUTION

A phylogeny of a set S of organisms is a graphical representation of the evolution of S, typically a rooted binary tree, leaf-labeled by S. However, events such as hybrid speciation and horizontal gene transfer require nontree models for accurate representations of evolution.

In what follows, we will assume that the individual gene datasets are recombination-free (so that meiotic recombination, or exchanges between sister chromosomes, does not take place); this simplifies our analysis, and allows us to assume that all gene evolution is tree-like (Gabriel *et al.*, 2002; Posada and Wiuf, 2003; Zhang and Jin, 2003). We also assume there are no gene gains or losses in the network.

It is clear that trees are inappropriate graphical models of *species* evolution when reticulation occurs, though still appropriate for *gene* evolution: in hybrid speciation, two lineages recombine to create a new species, as symbolized in Fig. 2(a), but genes evolve down trees contained in the network as shown in Figs. 2(b) and 2(c).

In lateral (i.e., horizontal) gene transfer, genetic material is transferred from one lineage to another without resulting in the production of a new lineage, as symbolized in Fig. 3(a). And, as in hybrid speciation, each site evolves down a tree within the network; that is, some sites are inherited through lateral transfer from another species, as in Fig. 3(b), while all others are inherited from the parent, as in Fig. 3(c).

3.1. Maddison's approach to phylogeny reconstruction

In 1997, Wayne Maddison (1997) made an important observation which directly suggests a technique for reconstructing phylogenetic networks, via a "separate analysis" approach, which we now describe. Maddison observed that when there is one reticulation in the network, there are two trees within the network, and every gene evolves down one of these two gene trees. More generally, Maddison suggested that

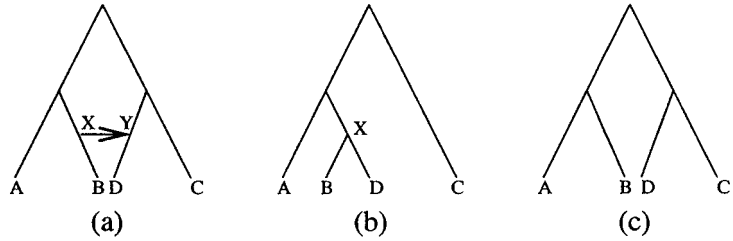

FIG. 3. Lateral gene transfer: the species network in (**a**) and its two induced (gene) trees in (**b**) and (**c**). The tree in (**b**) models the evolutionary history of those genes whose alleles in species D were horizontally transferred from X. The tree in (**c**) models the evolutionary history of all genes which evolved through lineal descent.

a network that contains multiple reticulations can be reconstructed from its constituent gene trees. However, given two gene trees, one can reconstruct a network with the smallest number of reticulations which induces both trees. Maddison's observations imply the following method for constructing phylogenetic networks:

- Step 1: For each gene dataset, infer a gene tree.
- Step 2: If the two trees are identical, return that tree. Else, find the minimum network that contains both trees.

While Maddison showed how to perform Step 2 when the minimum network contains a single reticulation, he left open how to do Step 2 when the network contains more than one reticulation, which is a computational limitation of Maddison's approach.

The other limitation is potentially more serious: if the gene trees have errors in them, then the minimal network that contains the gene trees may be incorrect. Therefore, Maddison's method needs to be modified to work with errors in the estimated gene trees.

In this paper, we address both problems. In Section 4, we show how to reconstruct a gt-network with any number of reticulations from accurate gene trees (under an additional assumption about the network). In Sections 5 and 6, we show how to reconstruct a network with a single reticulation from gene tree estimates that need not be accurate. In our future work, we will investigate how to combine these approaches.

4. RECONSTRUCTING GT-NETWORKS WHEN GENE TREE ESTIMATES ARE ACCURATE

There are two main limitations to Maddison's approach: (1) the construction of a network from two gene trees is only described explicitly when the network contains exactly one reticulation; (2) obtaining accurate binary trees in practice may not be possible in most cases. In this section, we address the first limitation by showing how to accurately construct a gt-network, with any number of reticulations, from two of its constituent gene trees. However, given two gene trees, the best we can hope for is to reconstruct the minimal gt-network that contains both trees. This is but a reflection of Occam's razor: in the absence of any additional biological information, infer the network with the minimum number of reticulation events that induces the gene trees. We address the second limitation in the next section.

We begin by characterizing networks in general, using the model of Moret *et al.* (2004), but with the added constraints that

C: there is at least one regular speciation event between any two reticulation events, and that a species does not become extinct immediately after a reticulation event.

Graph-theoretically, these constraints imply that there is at least one tree node (whose two children are also tree nodes) on the directed path between any two reticulation nodes, and that if one of the two children of a tree node is a reticulation node, then the other child is a tree node (see Moret *et al.* [2004] for a discussion of the ramifications of missing taxa on reconstructing networks). In this case, we can obtain the following result about the number of trees induced by, or contained inside, a network with m reticulations.

Theorem 1. *A species network N with m reticulation nodes induces 2^m distinct trees.*

Proof. We prove this by induction on m. It is easy to see that a network N with zero reticulations is a tree, and hence induces one tree. Further, a network N with one reticulation induces two trees that differ in the location of the subtree rooted at the reticulation node (see Figs. 2 and 3, for example). The only case where a network with one reticulation induces only one tree happens when the reticulation cycle contains only two nodes: the coalescent and reticulation nodes. However, in this case, network N violates the constraint **C** stated above.

Assume that any network with m reticulations induces 2^m trees and consider a network N with $m+1$ reticulations. Let x be a reticulation node in N below which there are no other reticulation nodes. Let u_1 and u_2 be the two parents of x. The node u_1 is a tree node and has another child v (a sibling of x), and

u_2 is a tree node and has another child w (different from v and a sibling of x). If we delete the edge (u_1, x), then the resulting network N' has m reticulations, and by the induction hypothesis, N' induces 2^m distinct trees, where the subtree rooted at x is attached to node u_2 in all these trees. If we delete the edge (u_2, x), then the resulting network N'' has m reticulations, and by the induction hypothesis, N'' induces 2^m distinct trees, where the subtree rooted at x is attached to node u_1 in all these trees. Hence, we have two sets of trees, each containing 2^m distinct trees, and clearly the two sets are different, due to the location of the subtree rooted at x. Again, the two sets of trees would be equivalent only in the case where the reticulation cycle of reticulation nodes x has (in addition to x) only one node, which is the coalescent node; however, in this case the network would violate the constraint **C** stated above. Therefore, we have 2^{m+1} distinct trees. ■

4.1. Efficient reconstruction of gt-networks from gene trees

Given a pair of trees induced by a gt-network, we reconstruct a minimal (in terms of the number of reticulation nodes) gt-network that induces these two trees. In what follows, we show how to efficiently reconstruct such a minimal network from a pair of trees. Hereafter, we use n to denote the number of leaves in the trees as well as in the networks.

4.1.1. Networks with a single reticulation event. In this section, we show how to construct a phylogenetic network with a single reticulation event (i.e., a single-gall gt-network) from two incongruent induced trees. Afterwards, we show how to extend the techniques to reconstruct gt-networks with multiple galls (reconstructing general networks with multiple reticulation events is still an open problem).

Theorem 2. *Let T_1 and T_2 be two incongruent trees induced by a network N with a single reticulation event. The network N can be reconstructed from T_1 and T_2 in $O(n)$ time.*

To prove this theorem, we need a series of auxiliary results.

Lemma 1. *Let T_1 and T_2 be two incongruent trees induced by a network N with a single gall Q_x^w. Then, $RP^Q(T_1) \subseteq U(T_1, T_2)$, and $RP^Q(T_2) \subseteq U(T_2, T_1)$.*

Proof. Let RP^Q be formed of the two paths p_1 and p_2 whose tail is w. Further, assume p_1 is the path attached to edge e_1 and p_2 is the path attached to e_2, where $RE(Q) = \{e_1, e_2\}$. Let X be the subtree rooted at node x, T_1 be obtained by removing edge e_1 from Q, and T_2 be obtained by removing edge e_2 from Q. Then, in T_1, the leaves of X are under the edges of p_2 but not under the edges of p_1, whereas in T_2, the leaves of X are under the edges of p_1 but not under the edges of p_2. Hence, the edges on $RP^Q(T_1)$ are incompatible with the edges of $RP^Q(T_2)$, and vice versa. Therefore, we have $RP^Q(T_1) \subseteq U(T_1, T_2)$ and $RP^Q(T_2) \subseteq U(T_2, T_1)$. ■

Definition 8. *Let T be a tree induced by a gt-network N. We denote by bf $NG(T)$ the set of all edges e that are not on any gall in N. Formally, $NG(T) = \{e \in E(T) : \text{for all galls } Q \text{ in } N, e \notin RP^Q(T)\}$.*

Lemma 2. *Let T_1 and T_2 be two incongruent trees induced by a single-gall gt-network N. Then, $NG(T_1) \cap U(T_1, T_2) = \emptyset$, and $NG(T_2) \cap U(T_2, T_1) = \emptyset$.*

Proof. Assume $e = (u, v)$ is an edge in $NG(T_1) \cap U(T_1, T_2)$. Let A be the subtree of T_1 rooted at v. Since $e \in U(T_1, T_2)$, then, for some edge $e' = (u', v')$ in T_2, $L(A) \cap L(B) \neq \emptyset$, where B is the subtree of T_2 rooted at v'. Let $X = L(A) \setminus (L(A) \cap L(B))$. Then, in tree T_1, X is under edge e, and in tree T_2, X is not under edge e'. Hence, edges e and e' are members of $E(Q)$ for some gall Q; a contradiction that $e \in NG(T_1)$. Therefore, $NG(T_1) \cap U(T_1, T_2) = \emptyset$; similarly, we prove that $NG(T_2) \cap U(T_2, T_1) = \emptyset$. ■

Proof of Theorem 2. From Lemmas 1 and 2, it follows that, if T_1 and T_2 are two incongruent trees induced by a single-gall gt-network N, then $U(T_1, T_2)$ forms a simple path p_1 in T_1 and $U(T_2, T_1)$ forms

a simple path p_2 in T_2. If this condition holds, we further test whether there exists a subtree T' whose root in T_1 is attached to an endpoint of p_1, and whose root in T_2 is attached to an endpoint of p_2. If so, T_1 and T_2 are induced by a gt-network N with a single gall Q_x^w, and there exists an edge in N from x to the root of T' (N is formed from T_1, by adding an edge to the root of T' from the other endpoint of p_1). If such T' does not exist, then the two trees are not induced by a single-gall gt-network. We now show how this can be achieved in $O(n)$ time.

Preprocess the trees so that the following pertains.

1. For every two leaves s_i and s_j in either tree, the least common ancestor (LCA) of these two leaves can be found in constant time. This can be achieved in $O(n)$ time using the techniques from Harel and Tarjan (1984) and Bender and Farach-Colton (2000).
2. For any internal node, v, the number $\beta(v)$ of leaves below v can be found in constant time. Further, if S_v is the set of leaves under v, then $LCA(S_v)$ can be found in constant time.

This can be achieved in $O(n)$ time using the techniques from Day (1985). After this preprocessing, computing $U(T_1, T_2)$ and $U(T_2, T_1)$ takes $O(n)$ time ($O(1)$ time for each edge, and there are $O(n)$ edges). This can be done by observing that an edge $e = (u, v)$ is in $U(T_1, T_2)$ if and only if $\beta(v) \neq \beta(LCA(S_v))$. It takes $O(n)$ time to check if $U(T_1, T_2)$ forms a simple path. Further, since there are at most four subtrees connected to the endpoints of each simple path in T_1 and T_1, it takes $O(n)$ time to test the existence of subtree T'; it is one of the (at most) four subtrees attached to the endpoints of the simple path. Hence, the single-gall gt-network can be reconstructed in $O(n)$ time. ■

4.1.2. gt-Networks with multiple reticulation events We now describe the more general case of multiple reticulation events, yet still for gt-networks. The intuition behind our algorithm is as follows. Given two trees T_1 and T_2 induced by a gt-network N, we first "mark" the edges of each tree that are incompatible with the other tree. If breaking differently m galls in N and similarly the remaining galls results in two trees T_1 and T_2, then $U(T_1, T_2)$ and $U(T_2, T_1)$ form m node-disjoint paths in each of the two trees T_1 and T_2, respectively.

While necessary, this condition is not sufficient; an extra step is needed, in which, for each maximal path p_1 of marked edges in T_1, there must exist a unique maximal path p_2 of marked edges in T_2, where the endpoints of the two paths correspond to one reticulation event.

Lemma 3. *Let T_1 and T_2 be two trees induced by a gt-network N. Further, assume that gall Q in N was broken in exactly the same way to obtain both T_1 and T_2. Then, $RP^Q(T_1) \cap U(T_1, T_2) = \emptyset$, and $RP^Q(T_2) \cap U(T_2, T_1) = \emptyset$.*

Proof. Since the gall Q is broken in exactly the same way to obtain the two trees T_1 and T_2, it follows that the edges on $RP^Q(T_1)$ induce the same bipartitions as those induced by the edges of $RP^Q(T_2)$. Hence, the edges of $RP^Q(T_1)$ and $RP^Q(T_2)$ are mutually compatible. Further, the edges of $RP^Q(T_1)$ are compatible with $E(T_2) \setminus RP^Q(T_2)$; otherwise, the network N would not be a gt-network (there would be two "overlapping" galls). Similarly, the edges of $RP^Q(T_2)$ are compatible with $E(T_1) \setminus RP^Q(T_1)$. Therefore, it follows that $RP^Q(T_1) \cap U(T_1, T_2) = \emptyset$ and $RP^Q(T_2) \cap U(T_2, T_1) = \emptyset$. ■

Theorem 3. *Let N be a gt-network with q galls $\{Q_1, Q_2, \ldots, Q_q\}$, and T_1 and T_2 be two trees induced by N. Further, assume that exactly m of the q galls were broken in the two possible ways to obtain the two trees T_1 and T_2, and the other $q - m$ galls were each broken similarly. Then, $U(T_1, T_2)$ forms m node-disjoint undirected paths in T_1, and $U(T_2, T_1)$ forms m node-disjoint undirected paths in T_2.*

The proof follows immediately from Lemma 1, Lemma 3, and Lemma 2, and is omitted.

Let T_1 and T_2 be two trees induces by a gt-network N, such that $U(T_1, T_2)$ forms a set of node-disjoint undirected paths in T_1, and $U(T_2, T_1)$ forms a set of node-disjoint undirected paths in T_2. Let p_1 be one such path in $U(T_1, T_2)$, and p_2 be one such path in $U(T_2, T_1)$. Further, assume $END(p_1) = (U_1, V_1)$ and $END(p_2) = (U_2, V_2)$.

Definition 9. *We say that* p_1 yields p_2 in one rSPR move *(via subtree X), denoted* $p_1 \models^X p_2$, *if there exists a nonempty subtree X such that*

1. *X is a subtree of either* U_1 *or* V_1,
2. *X is a subtree of either* U_2 *or* V_2, *and*
3. $p_1 \cap U(T_1', T_2') = \emptyset$ *and* $p_2 \cap U(T_2', T_1') = \emptyset$, *where* $T_1' = T_1 \setminus X$ *and* $T_2' = T_2 \setminus X$.

Theorem 4. *Let* T_1 *and* T_2 *be two trees induced by a gt-network N. Further, assume that* $U(T_1, T_2)$ *forms a set* P_1 *of m node-disjoint undirected paths* $p_1^1, p_2^1, \ldots, p_m^1$ *in* T_1, *and* $U(T_2, T_1)$ *forms a set* P_2 *of m node-disjoint undirected paths* $p_1^2, p_2^2, \ldots, p_m^2$ *in* T_2. *Then, the smallest number of galls in N is m if there is an injective function* $f : P_1 \to P_2$ *and m subtrees* $X_1, X_2, \ldots, X_m$ *such that* $f(p_i^1) = p_j^2$ *iff* $p_i^1 \models^{X_i} p_j^2$, *where* $1 \leq i, j \leq m$.

Proof. Let P_1 and P_2 be the two sets of paths in the lemma, and let f be the injective function. Let $p_i^1 \in P_1$ and $p_j^2 \in P_2$ be two paths such that $f(p_i^1) = f(p_j^2)$. Assume X_i is the subtree such that $p_i^1 \models^{X_i} p_j^2$. Then, X_i is the subtree whose pruning from T_1 and regrafting it to another edge (to obtain tree T_2) yielded paths p_i^1 and p_j^2 in the two trees, respectively. Since there are m such pairs of paths, there are m such subtrees X_i whose pruning and regrafting in T_1 would yield a tree T such that $U(T_1, T_2) = \emptyset$, which implies $T = T_1$. Hence, the smallest number of galls in N is m. ■

The result we obtained in Theorem 4 is related to results obtained by Gusfield and Hickerson (2004) and Bafna and Bansal (2004) concerning the minimum number of recombination events needed to explain the evolutionary history of a set of sequences

Theorem 5. *Let* T_1 *and* T_2 *be two binary trees on n leaves. We can decide whether there exists a gt-network inducing* T_1 *and* T_2 *and then construct a minimal such gt-network, in* $O(mn)$ *time, where m is the smallest number of reticulations in any gt-network containing the two trees.*

Proof. Using the techniques described in the proof of Theorem 2, we can compute $U(T_1, T_2)$, $U(T_2, T_1)$, check whether they form simple paths, and test for the conditions of Theorems 3 and 4, all in $O(n)$ time. We can find $f(p_i^1)$, if it exists, in $O(1)$ time, by using the "highest" node of path p_i^1 and finding its counterpart in T_2; to find that subtree X_i such that $p_i^1 \models^X p_j^2$, we need to compute the four pairwise intersections of a set in $END(p_i^1)$ and a set in $END(p_j^2)$, each of which takes $O(n)$ time, using bit vector representation of sets. Hence, we can decide whether T_1 and T_2 are induced by a gt-network, with minimal number m of reticulation events (galls), in $O(mn)$ time.

By Theorem 5, we can find the m subtrees $\{X_1, \ldots, X_m\}$, such that $p_i^1 \models^{X_i} p_i^2$, for $1 \leq i \leq m$, where p_i^1 is a path in $U(T_1, T_2)$, p_i^2 is a path in $U(T_2, T_1)$, and $f(p_i^1) = p_i^2$ (f is the injective function in the definition of $\models^X$). All this can be done in $O(mn)$ time. We form the gt-network N from T_1 as follows. For each path p_i^1 in $U(T_1, T_2)$ and its corresponding subtree X_i, X_i will be attached to one end of p_i^1. We add another edge from the other end of p_i^1 to the root of X_i, thus creating a network N with m galls. Since the paths are node-disjoint, N will be a gt-network. ■

5. RECONSTRUCTING GT-NETWORKS WHEN GENE TREE ESTIMATES ARE INACCURATE

The main limiting factor in Maddison's approach is that methods, even if statistically consistent, can fail to recover the true tree. Even on quite long sequences, some topological error is often present. This topological error can be tolerated in a phylogenetic analysis, but it makes the inference of phylogenetic networks from constituent gene trees difficult. To overcome these limits, we propose a method that allows for error in the estimates of the individual gene trees; consequently, our method performs much better in practice (as our simulation studies show).

Before we describe the method, we provide some insight into its design. When methods such as maximum parsimony or maximum likelihood are used to infer trees, typically many trees are returned, rather than a single best tree. For example, in maximum parsimony searches, especially with larger datasets, there are often many equally good trees (all having the same best score), and all can be returned (along with suboptimal trees, if desired). In maximum likelihood, although the best-scoring tree may be unique, the difference in quality between that tree and the next best tree(s) can be statistically insignificant, and so again, a number of trees can be returned (Shimodaira and Hasegawa, 1999). A common output of a phylogenetic analysis is the strict consensus of these trees (that is, the most resolved common contraction of all the best trees found).

The interesting, and highly relevant, point here is the following observation, supported by both empirical studies on real datasets and simulations: *the strict consensus tree will often be a contraction of the true tree*. Thus, even when every tree in the set of best trees is a little bit wrong, the strict consensus tree (which contains only those edges common to all the best trees) is likely to be a contraction of the true tree. This observation suggests the following approach to inferring phylogenetic networks.

Proposed Approach

- **Step 1:** For each gene dataset, use a method (such as maximum parsimony or maximum likelihood) of choice, to construct a set of "best" trees, thus producing sets $\mathcal{T}_1$ and $\mathcal{T}_2$.
- **Step 2:** Compute the strict consensus tree t_i for $\mathcal{T}_i$, for $i = 1, 2$.
- **Step 3:** Find trees T_1 and T_2 refining t_1 and t_2 such that T_i refines t_i for each $i = 1, 2$, and T_1 and T_2 are induced trees within a gt-network with p reticulations, for some minimum p.

When $p = 0$, the two consensus trees are compatible, and we would return the compatibility tree; see Section 2.1. We now show how to handle the third step in this method when $p = 1$ (solving this for general p is currently an open problem). In this case, Step 3 involves solving the following problem.

- **Combining consensus trees into a network** (the ConsTree-Network Problem)
  Input: Two trees, t_1 and t_2, on the same set of leaves (not assumed to be binary).
  Output: A network N inducing trees T_1 and T_2, such that N contains one reticulation, and T_i refines t_i, for $i = 1, 2$, if it exists; else *fail*.

We now provide a linear-time algorithm for this problem. There are two cases to consider: when the two consensus trees are compatible, and when the two trees are incompatible.

5.1. Compatible consensus trees

In most cases, if the consensus trees share a common refinement, we might believe the evolution to be tree-like (in which case we should combine the datasets and analyze a tree directly). However, suppose we have reason to believe that a dataset has undergone reticulation, so that a tree is not an appropriate representation of the true tree. In this case, we can still seek reticulate evolutionary scenarios compatible with our observations. We begin with a simple lemma.

Observation 1. *Let T be a binary tree that refines an unresolved tree t, and let p be a path in tree T. Then, when restricted to the edges of t, p forms a path in t as well.*

Lemma 4. *Let t be an unresolved tree. Then, there exist two distinct binary trees T_1 and T_2 that refine t and such that T_1 and T_2 are induced by a network with a single reticulation.*

Proof. Let x be a node with outdegree 3, and let v_1, v_2, and v_3 be the three children of node x. We obtain T_1 from t by removing the edges (x, v_1) and (x, v_2), adding a new node u with an edge (x, u), and then making v_1 and v_2 children of u. The tree T_2 can be obtained from t by removing the edges (x, v_2)

and (x, v_3), adding a new node u with an edge (x, u), and then making v_2 and v_3 children of u. The rest of the nodes of T_1 and T_2 are resolved identically in both trees. It is obvious that T_2 can be obtained from T_1 by pruning v_2 from its parent and attaching it to edge (x, v_3) in T_1, and hence T_1 and T_2 are induced by a network with one reticulation. ■

Lemma 4 can be generalized to the case where t_1 and t_2 are two unresolved, yet compatible trees, as follows.

Lemma 5. *Let t_1 and t_2 be two compatible unresolved trees. Then, there exist two binary trees T_1 and T_2 that refine t_1 and t_2, respectively, and T_1 and T_2 are induced by a network with a single reticulation if and only if t_1 and t_2 have a common refinement t that is not fully resolved. Furthermore, we can determine whether these two trees exist, and construct them, in $O(n)$ time.*

Proof. The proof of the "if" part follows from Lemma 4. We prove the "only if" part. Let T_1 and T_2 be two binary trees that refine two unresolved (compatible) trees t_1 and t_2, such that T_1 and T_2 are induced by a network with a single reticulation. Since t_1 and t_2 are compatible, then they share a common refinement t. The two binary trees T_1 and T_2 also refine t. Since T_1 and T_2 are different binary trees and refine the same tree t, it follows that t is not fully resolved. ■

5.2. Incompatible consensus trees

We now address the last remaining case, where the consensus trees are incompatible. We begin with a simple lemma.

Lemma 6. *Let T_1 and T_2 be two binary trees that refine two unresolved trees t_1 and t_2. Then, $U(t_1, t_2) \subseteq U(T_1, T_2)$.*

Proof. Let $e \in U(t_1, t_2)$. Then, $e \in E(t_1)$, and consequently $e \in E(T_1)$. Further, e is incompatible with t_2 and hence is incompatible with T_2. It follows that $e \in U(T_1, T_2)$. Therefore, $U(t_1, t_2) \subseteq U(T_1, T_2)$. ■

Lemma 7. *Let t_1 and t_2 be two unresolved incompatible trees. If there exist two binary trees T_1 and T_2 that refine t_1 and t_2, respectively, and such that T_1 and T_2 are induced by a network with a minimum of one reticulation, then $U(t_1, t_2)$ and $U(t_2, t_1)$ are both simple paths in t_1 and t_2, respectively.*

Proof. Assume $U(t_1, t_2)$ is not a simple path in t_1. Then, by Theorem 3, it follows that $U(T_1, T_2)$ is not a simple path in T_1, and hence T_1 and T_2 cannot be induced by a network with a minimum of one reticulation; a contradiction. Therefore, $U(t_1, t_2)$ forms a simple path in t_1. Similarly, we establish that $U(t_2, t_1)$ forms a simple path in t_2. ■

Lemma 8. *Let t_1 and t_2 be two incompatible unresolved trees, such that $U(t_1, t_2)$ forms a path p_1 in t_1 and $U(t_2, t_1)$ forms a path p_2 in t_2. Further, let $END(p_1) = (A_1, B_1)$ and $END(p_2) = (A_2, B_2)$. Let X_i, $1 \leq i \leq 4$, be the following four sets: $X_1 = (A_1 - A_2) \cap (B_2 - B_1)$, $X_2 = (A_1 - B_2) \cap (A_2 - B_1)$, $X_3 = (B_1 - A_2) \cap (B_2 - A_1)$, and $X_4 = (B_1 - B_2) \cap (A_2 - A_1)$. Then, there exist two binary trees T_1 and T_2 that refine t_1 and t_2, respectively, and T_1 and T_2 are induced by a network with a minimum of one reticulation, if and only if there exists an i, $1 \leq i \leq 4$, such that*

(**C1**) *$t_1|_{S \setminus X_i}$ and $t_2|_{S \setminus X_i}$ are compatible,*
(**C2**) *$t_1|_{X_i}$ and $t_2|_{X_i}$ are compatible, and*
(**C3**) *$t_1|_{S \setminus X_i}$ contains all the edges in $U(t_1, t_2)$, and $t_2|_{S \setminus X_i}$ contains all the edges in $U(t_2, t_1)$.*

Proof. Assume that X_i, for some $1 \leq i \leq 4$, satisfies both conditions C1 and C2. Then, resolve $t_1|_{S \setminus X_i}$ and $t_2|_{S \setminus X_i}$ identically, resolve $t_1|_{X_i}$ and $t_2|_{X_i}$ identically, and finally attach the resolved subtrees $t_1|_{X_i}$ and $t_2|_{X_i}$ in their corresponding subtrees. The result is obviously two binary trees T_1 and T_2 that differ only in the location of the subtree leaf-labeled by X_i; i.e., T_1 and T_2 are induced by a network with one

reticulation. Let T_1 and T_2 be two binary trees that resolve the two incompatible unresolved trees t_1 and t_2, such that T_1 and T_2 are induced by a network with one reticulation. By Lemma 6, $p_1 \subseteq U(T_1, T_2)$ and $p_2 \subseteq U(T_2, T_1)$. By Theorem 4, T_2 can be obtained from T_1 by pruning a subtree t' from one side of the path $U(T_1, T_2)$ and regrafting it on the other side of the path. It follows that $t_1|_{S\setminus L(t')}$ and $t_2|_{S\setminus L(t')}$ are compatible and also that $t_1|_{L(t')}$ and $t_2|_{L(t')}$ are compatible (since they refine the same tree t'). It is straightforward to verify that $L(t')$ is equal to X_i, for some $1 \leq i \leq 4$. ■

We now state the major theorem of this section.

Theorem 6. *We can solve the ConsTree-Network Problem in $O(n)$ time. That is, given two unresolved trees t_1 and t_2, in $O(n)$ time we can find two binary trees T_1 and T_2 that refine t_1 and t_2, respectively, such that T_1 and T_2 are induced by a network with one reticulation, when such a pair of trees exist. Further, once we have T_1 and T_2, we can compute a phylogenetic network with exactly one reticulation event inducing these trees in $O(n)$ additional time.*

Proof. Using the same techniques as in the proof of Theorem 2, we can compute $U(t_1, t_2)$ and $U(t_2, t_1)$ in $O(n)$ time. Then, we check whether conditions C1 and C2 of Lemma 8 hold; if so, then we can obtain two binary trees T_1 and T_2 that resolve t_1 and t_2, such that T_1 and T_2 are induced by a network with one reticulation. Having preprocessed the trees, testing the conditions of these two lemmas can be achieved in $O(n)$ time. Using bit vectors to represent the sets of taxa, we can preprocess the trees in $O(n)$ time such that we store at each node the set of taxa under it; hence, it takes $O(n)$ time to compute the sets X_i, $1 \leq i \leq 4$. We construct N from T_1 and T_2 in $O(n)$ time using Theorem 5. Hence, the algorithm takes $O(n)$ time. ■

6. SPNET: OUR TECHNIQUE FOR INFERRING GT-NETWORKS

SpNet, for Species Network, is a method we have designed for inferring networks (or trees, depending on the data) under realistic conditions. We base SpNet on the approach we outlined in the previous section, but we specifically use maximum likelihood for tree reconstruction, and we compute the strict consensus of the best two trees for each dataset. In order to facilitate a comparison to other methods, such as NeighborNet, we do not allow SpNet to return "fail," and so we apply Neighbor Joining (NJ) to all inputs on which we would otherwise return "fail."

SpNet

- **Step 1:** We find the best two trees on each dataset under maximum likelihood.
- **Step 2:** For each dataset, we compute the strict consensus of the two trees, thus producing the trees t_1 and t_2.
- **Step 3:** If t_1 and t_2 are compatible, we combine datasets and analyze the combined (i.e., concatenated) dataset using NJ, thus returning a tree. Else, we apply our algorithm for ConsTree-Network to t_1 and t_2. If we can, we return a network N with one reticulation (if trees T_1 and T_2 exist refining t_1 and t_2, respectively, contained within the network N); if no such network exists, we apply NJ to the concatenated dataset and return a tree. (Alternatively, we could simply return "fail.")

7. EXPERIMENTAL EVALUATION

In this section, we evaluate (using simulations) the performance of three methods (SpNet, NeighborNet [Bryant and Moulton, 2002], and neighbor joining [Saitou and Nei, 1987]) and report on our findings.

7.1. Experimental settings

7.1.1. Methods. **Neighbor joining (NJ).** (Saitou and Nei, 1987) Neighbor Joining is one of the most popular distance-based methods. NJ takes a distance matrix as input and outputs a tree. For every two

taxa, it determines a score, based on the distance matrix. At each step, the algorithm joins the pair with the minimum score, making a subtree whose root replaces the two chosen taxa in the matrix. The distances are recalculated to this new node, and the "joining" is repeated until only three nodes remain. These are joined to form an unrooted binary tree. NJ takes $O(n^3)$ time, and we used PAUP* to run the method.

NeighborNet (NNet). (Bryant and Moulton, 2002) NeighborNet is an $O(n^3)$ method for constructing phylogenetic networks. Like NJ, it iteratively selects pairs of taxa to group together, but it does not join them immediately. Rather, at a later stage, it agglomerates pairs of pairs which share one node in common. NeighborNet generates a *circular split system* (Bandelt and Dress, 1992) rather than a hierarchy or a tree, which can subsequently be represented by a planar *split graph* (Dress and Huson, 1998). In these graphs, bipartitions of *splits* of the taxa are represented by classes of parallel lines, and conflicting signals or incompatibilities appear as boxes. The authors of NeighborNet suggest this method can be used to detect complex evolutionary processes like recombination, lateral transfer, and hybridization (Bryant and Moulton, 2002). We used the Linux 1.2 version of the NeighborNet software package.

SpNet. We implemented our method, SpNet, which is outlined in Section 6.

7.1.2. Tools and measures. We used the tools of Nakhleh *et al.* (2003) to generate random networks as well as to simulate the evolution of DNA sequences down those networks. To quantify the error rate of methods and compare their topological accuracy, we used split-based false positive and false negative rates. The reason behind choosing split-based comparison (rather than tree-based or tripartition-based) is that the output of NeighborNet is a set of splits, rather than a phylogenetic network. Further, split-based comparison of phylogenetic networks induces a metric, whereas that is not the case for general networks (Nakhleh *et al.*, 2004).

Topological accuracy. Let N be a phylogenetic network whose set of induced trees is denoted by $\mathcal{T}(N)$. We define the set of splits of N, denoted by $C(N)$, to be

$$C(N) = \cup_{T \in \mathcal{T}(N)} C(T).$$

Given a model network N_1 and an inferred network N_2, we define the *false positive rate* (FP) and *false negative rate* (FN) as follows:

$$FP(N_1, N_2) = \frac{|C(N_2) - C(N_1)|}{|C(N_1)|}, \qquad FN(N_1, N_2) = \frac{|C(N_1) - C(N_2)|}{|C(N_1)|}.$$

Note that both quantities are normalized by the number of splits in the model phylogeny.

7.1.3. Simulation parameters.

Diameters. We generated random networks with diameter 2 (where the diameter is the maximum expected number of changes of a random site on any leaf-to-leaf path). We then scaled the edge lengths by scaling factors of 0.05, 0.1, and 0.25, producing networks with diameters of 0.1, 0.2, and 0.5, respectively.

Deviation factor. To deviate the networks from ultrametricity, we used a deviation factor 2: for each edge, we used a random variate x in the range $[-\ln 2, \ln 2]$ and multiplied the edge length by e^x. This produces a fairly modest deviation from the molecular clock.

Numbers of hybrids. NNet handles any number of reticulation events, whereas SpNet, at this stage, handles only the cases of 0 and 1 reticulations. Due to this limitation, we generated only networks with 0, 1, and 2 hybrids.

Numbers of taxa. We generated networks with 10 and 20 leaves, one network for each combination of diameter, number of taxa, and number of hybrids.

Rooting the networks. Our method, SpNet, assumes rooted trees and networks. To obtain rooted networks and trees, we used an outgroup in each network.

Sequences. We then evolved sequences on these networks using the GTR+Γ + I (gamma distributed rates, with invariable sites) model of evolution. We used the parameter settings of Zwickl and Hillis (2002), which are the following.

- Shape parameter for Γ: 0.8168
- Proportion of invariable sites: 0.1
- Base frequencies: A:0.1776, C:0.3336, G:0.2595, and T:0.2293
- Rate matrix: A→C:3.297 A→G:12.55 A→T:1.167 C→G:2.060 C→T:13.01 G→T:1.00

We generated random sequences of length 500, 1,000, 2,000, and 4,000.

Numbers of datasets. For each combination of sequence length and network settings, we generated 25 sets of sequences.

Software. We used PAUP* (Swofford, 1996) for NJ and ML, and the NeighborNet software package (the linux 1.2 version); we implemented our method, SpNet, using C++ and the LEDA library of data structures and algorithms (*www.algorithmic-solutions.com/enleda.htm*).

Obtaining the ML trees in PAUP*. To obtain the best 100 ML trees, we first ran a quick parsimony search to get 100 initial trees, and then a search (using TBR branch swapping) was used to get to the best possible trees with respect to ML. We specified the correct model parameters for the ML search. The search was limited to five minutes for each dataset. The following is the PAUP block we used.

```
set monitor=yes maxtrees=100 increase=no criterion=parsimony;
hsearch start=stepwise addseq=random nreps=100 swap=tbr hold=1
nchuck=1 chuckscore=1;
filter best=yes;
set criterion=likelihood maxtrees=100 increase=no;
lset nst=6 rmatrix=(3.297 12.55 1.167 2.060 13.01)
basefreq=(0.1776 0.3336 0.2595) rates=gamma shape=0.8168
Pinvar=0.1 rescale=100;
hsearch start=current swap=tbr nchuck=100 nbest=100 chuckscore=no
timelimit=300;
```

7.2. Experimental results

We have done extensive studies evaluating the performance of the three methods. We focus here on the results of our experiments on 20-taxon trees and networks with one reticulation event. False negative and false positive rates below 10% are good, with rates below 5% very good, in evaluating tree reconstruction methods.

All three methods have excellent false negative rates (less than 5%) on trees and very good false negative rates (less than 10%) on networks with one reticulation, when given long enough sequences.

However, the methods can be clearly distinguished in terms of their false positive rates; see Fig. 4. NeighborNet has very poor false positive rates on both trees and networks with one reticulation, even at very long sequences (4,000 nucleotides), while SPNET has a very low false positive rate.

Since SPNET uses NJ to analyze datasets whenever it cannot infer a network with a single reticulation, a comparison between SPNET and NJ is worth making. On trees they have essentially identical performance, as expected. On networks with a single reticulation, however, their performance is distinguishable: SPNET has an almost 0% false positive rate, which means that it produces a network with essentially no false edges, while NJ has (in these experiments) a false positive rate that is approximately 5%. The two methods have very close false negative rates. Thus, on networks with one reticulation, SPNET produces networks which are, with some reliability, *contractions* of the true network, while NJ's performance does not have the same reliability.

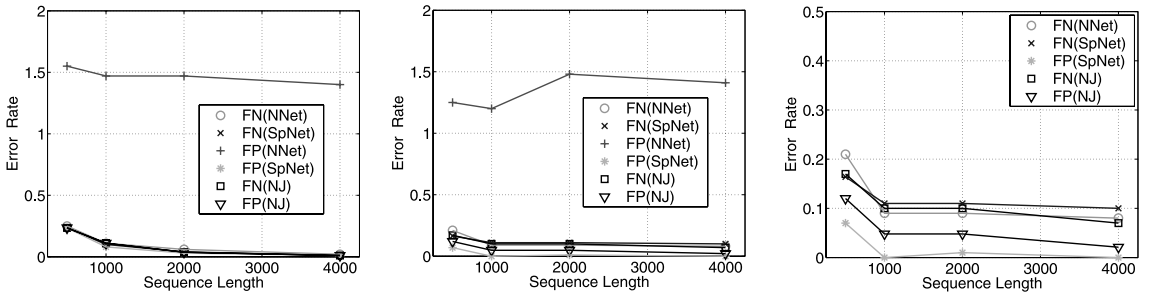

FIG. 4. FN and FP error rates of NeighborNet (NNet) and SPNET on 20-taxon networks, with 0.1 scaling factor, with tree model phylogeny (**left**), and 1-hybrid network (**middle**). The **rightmost** graph shows the results without the FP rate of NNet.

8. CONCLUSIONS

Our experiments show that NeighborNet, the current most commonly used method for network reconstruction using a "combined analysis" approach, has poor performance with respect to its false positive rate; we hypothesize that this phenomenon is likely to be true of combined analysis approaches in general. Our new method, SPNET, works better than NeighborNet and NJ in terms of reconstructing phylogenetic networks with a single reticulation.

The main open problem is to develop methods which can accurately reconstruct networks, in general, with more than one reticulation. In our future research, we plan to combine and extend the techniques we developed in this paper in order to develop robust methods for estimating phylogenetic networks with many reticulations. An obvious direction is to solve the following problem: *given two (or more) nonbinary trees on the same set of taxa, find the minimum network that contains refinements of each of the trees.* The quality of our method on networks with one reticulation suggests that a solution to this problem will be very useful for phylogenetic network reconstruction and should have better accuracy (with respect to false positives) than existing approaches. However, the problem remains of unknown computational complexity, even for gt-networks.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under grants DEB 01-20709 (Linder and Warnow), EIA 01-21651 (Warnow), EIA 01-21680 (Linder and Warnow), and EF 01-31453 (Linder and Warnow), by the David and Lucile Packard Foundation (Warnow), by the Institute for Cellular and Molecular Biology at UT-Austin (Warnow), by the Program in Evolutionary Dynamics at Harvard University (Warnow), and by the Radcliffe Institute for Advanced Study (Warnow).

REFERENCES

Bafna, V., and Bansal, V. 2004. The number of recombination events in a sample history: Conflict graph and lower bounds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1(2), 78–90.

Bandelt, H.J., and Dress, A.W.M. 1992. Split decomposition: A new and useful approach to phylogenetic analysis of distance data. *Mol. Phyl. Evol.* 1, 242–252.

Bender, M.A., and Farach-Colton, M. 2000 The LCA problem revisited. *Latin American Theoretical INformatics*, 88–94.

Bryant, D., and Moulton, V. 2002. NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. *Proc. 2nd Workshop Algorithms in Bioinformatics (WABI '02)*, 375–391. vol. 2452 of Lecture Notes in Computer Science.

Day, W.H.E. 1985. Optimal algorithms for comparing trees with labeled leaves. *J. Classification* 2, 7–28.

Dress, A., and Huson, D. 1998. Computing phylogenetic networks from split systems. Manuscript.

Gabriel, S.B., Schaffner, S.F., Nguyen, H., Moore, J.M., Roy, J., Blumenstiel, B., Higgins, J., DeFelice, M., Lochner, A., Faggart, M., Liu-Cordero, S.N., Rotimi, C., Adeyemo, A., Cooper, R., Ward, R., Lander, E.S., Daly, M.J., and Altshuler, D. 2002. The structure of haplotype blocks in the human genome. *Science* 296(5576), 2225–2229.

Gusfield, D. 1991. Efficient algorithms for inferring evolutionary trees. *Networks* 21, 19–28.

Gusfield, D., Eddhu, S., and Langley, C. 2003. Efficient reconstruction of phylogenetic networks with constrained recombination. *Proc. Computational Systems Bioinformatics (CSB '03)*.

Gusfield, D., and Hickerson, D. 2004. A new efficiently-computed lower bound on the number of recombinations needed in phylogentic networks. Technical Report CSE-2004-6, University of California, Davis.

Harel, D., and Tarjan, R.E. 1984. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* 13(2), 338–355.

Lawrence, J.G., and Ochman, H. 2002. Reconciling the many faces of lateral gene transfer. *Trends Microbiol.* 10, 1–4.

Maddison, W.P. 1997. Gene trees in species trees. *Systematic Biol.* 46(3), 523–536.

Moret, B.M.E., Nakhleh, L., Warnow, T., Linder, C.R., Tholse, A., Padolina, A., Sun, J., and Timme, R. 2004. Phylogenetic networks: Modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1(1), 13–23.

Nakhleh, L., Clement, A., Warnow, T., Linder, C.R., and Moret, B.M.E. 2004. Quality measures for phylogenetic networks. Technical Report TR-CS-2004-06. University of New Mexico, Albuquerque, New Mexico.

Nakhleh, L., Sun, J., Warnow, T., Linder, C.R., Moret, B.M.E., and Tholse, A. 2003. Towards the development of computational tools for evaluating phylogenetic network reconstruction method. *Proc. 8th Pacific Symp. on Biocomputing (PSB '03)*, 315–326.

Page, R., and Charleston, M.A. 1998. Trees within trees: Phylogeny and historical associations. *Trends Ecol. Evol.* 13, 356–359.

Posada, D., and Wiuf, C. 2003. Simulating haplotype blocks in the human genome. *Bioinformatics* 19(2), 289–290.

Saitou, N., and Nei, M. 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* 4, 406–425.

Shimodaira, H., and Hasegawa, M. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Mol. Biol. Evol.* 16, 1114–1116.

Swofford, D.L. 1996. *PAUP*: Phylogenetic analysis using parsimony (and other methods)*, version 4.0, Sinauer Associates, Underland, MA.

Wang, L., Zhang, K., and Zhang, L. 2001. Perfect phylogenetic networks with recombination. *J. Comp. Biol.* 8(1), 69–78.

Warnow, T. 1994. Tree compatibility and inferring evolutionary history. *J. Algorithms* 16, 388–407.

Zhang, K., and Jin, L. 2003. HaploBlockFinder: Haplotype block analyses. *Bioinformatics* 19(10), 1300–1301.

Zwickl, D., and Hillis, D. 2002. Increased taxon sampling greatly reduces phylogenetic error. *Systematic Biol.* 51(4), 588–598.

Address correspondence to:
Luay Nakhleh
Department of Computer Science
Rice University
6100 Main Street, MS 132
Houston, Texas 77005-1892

E-mail: nakhleh@cs.rice.edu