Reconstructing Reticulate Evolution in Species – Theory and Practice

Luay Nakhleh
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
nakhleh@cs.utexas.edu

Tandy Warnow*
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
tandy@cs.utexas.edu

C. Randal Linder
Section of Integrative Biology
School of Biological Sciences
University of Texas at Austin
Austin, Texas 78712
rlinder@mail.utexas.edu

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 network are node-disjoint ("galled"); we prove accuracy guarantees for our first method by presenting a formal characterization of the set of gene trees defined by a species network. 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.

Categories and Subject Descriptors

F.2.0 [**Theory of Computation**]: General; G.4 [**Mathematical Software**]: Algorithm Design and Analysis; J.3 [**Computer Applications**]: Life and Medical Sciences

General Terms

Algorithms, Theory, Experimentation.

Keywords

Phylogenetic networks, gene trees, subtree prune and regraft.

*Currently on sabbatical at the Radcliffe Institute for Advanced Study.

RECOMB'04, March 27–31, 2004, San Diego, California, USA.

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 [8] 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 exist for reconstructing phylogenetic networks from gene datasets; of these, NeighborNet by Bryant and Moulton [2] and a method by Wayne Maddison [10] 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. Maddison also showed the connection between phylogenetic network reconstruction and the calculation of *rooted subtree prune and regraft* (rSPR) distances (which we define and discuss below). Maddison showed explicitly how to construct a network containing a single reticulation from its two trees (which are related by one rSPR move), and suggested that networks with additional reticulations could be inferred from individual gene trees, provided that the rSPR distance between two constituent gene trees could be calculated. However, Maddison did not show (or prove) how to use rSPR distance calculations in order to construct phylogenetic networks. Furthermore, the rSPR Distance problem is currently of unknown complexity [1] (a naïve algorithm would solve the problem in $O(n^{2m})$ time, where n is the number of leaves in each of the trees, and m is the rSPR distance between the trees).

In this paper we consider the inference of "gt-networks" (for "galled tree" networks, which is the terminology used in [6]); these are phylogenetic networks in which reticula-

tion events are constrained so as to be evolutionarily independent of each other (see [6] for a biological justification of the model). This model was first introduced by Wang *et al.* [15], and later formalized and further pursued by Gusfield *et al.* [6]. For this special case, 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 and the rSPR operation. 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 a formal characterization of gene trees within species networks in terms of the rSPR distances among them, and then 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 nice 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, and we (briefly) 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 7 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. 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, and 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, and 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 $\pi(e_2)$. This definition of compatibility naturally extends to sets of bipartitions, and hence also to 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 and strict consensus trees can be found in $O(kn)$ time, where there are k trees on the same set of n leaves [5, 16, 3].

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 in [9], 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 co-exist 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 co-exist 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 in [10, 12, 9].

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.* [15] and Gusfield *et al.* [6].

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".*

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 3. *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 4. *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 Figure 1(a). To obtain the tree in Figure 1(b), the gall was broken by removing edge (u_1, x) and applying forced contraction to node x; to obtain the tree in Figure 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 5. *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). 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 Figure 1(b) form the reticulation path $RP^Q(T_1)$, and the marked edges in tree T_2 of Figure 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).

2.4 The rSPR operation

The rSPR operation transforms rooted trees into other rooted trees, and gene trees contained inside species networks are related to each other by rSPR operations; we now explain this relationship. Observe that the two rooted trees T_1 and T_2 in Figures 1(b) and 1(c) (induced by the network in Figure 1(a)) differ only in the location of the subtree T. Tree T_2 can be obtained from T_1 by "pruning" the subtree T and "regrafting" it to another edge. In this case, we say that T_2 is obtained from T_1 by one *rooted subtree prune and regraft* (rSPR) operation.

DEFINITION 6. *(From [1]) A* **rooted subtree prune and regraft** *(rSPR) on a rooted binary tree T is defined as cutting any edge and thereby pruning a subtree, t, and then regrafting the subtree by the same cut edge to a new vertex obtained by subdividing a pre-existing edge in $T - t$. We also apply a forced contraction to maintain the binary property of the resulting tree.*

The rSPR distance between two rooted binary trees T_1 and T_2, denoted by $d_{rSPR}(T_1, T_2)$, is the minimum number of rSPR operations needed to obtain T_2 from T_1. Computing the rSPR distance between trees plays a central role in the method that we propose for reconstructing networks. We formally define the (decision) rSPR distance problem as follows.

DEFINITION 7. *(The m-rSPR Distance Problem)*

Input: *Two binary trees, T_1 and T_2, leaf-labeled by a set S of n taxa, and a nonnegative integer m.*

Question: *Is $d_{rSPR}(T_1, T_2) = m$?*

The m-rSPR Distance Problem is of unknown computational complexity [1]. However, for any constant m, we can solve the m-rSPR Distance Problem in polynomial-time, as we show in the following theorem.

THEOREM 1. *Given two binary trees T_1 and T_2 leaf-labeled by a set S of n leaves, and a constant m, we can decide whether $d_{rSPR}(T_1, T_2) = m$ in $O(n^{2m})$ time.*

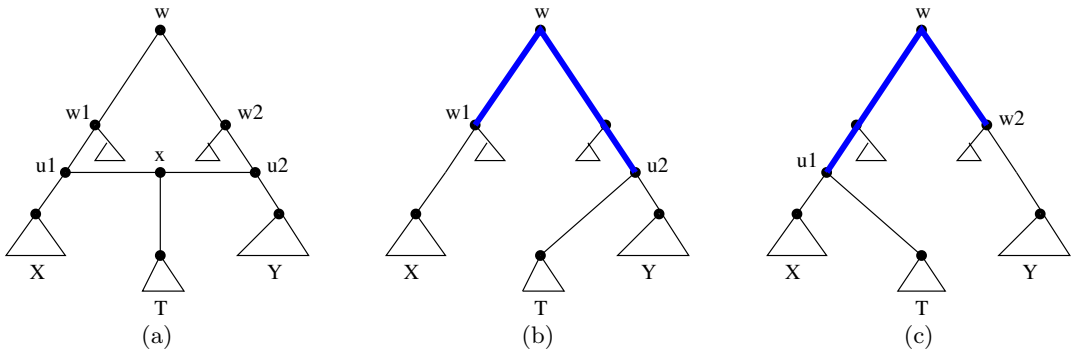

Figure 1: (a) A gall Q whose coalescent and reticulation nodes are w and x respectively. (b) and (c) show the two possible ways of "breaking" the gall Q to induce trees T_1 and T_2, respectively. The marked edges in T_1 and T_2 form $RP^Q(T_1)$ and $RP^Q(T_2)$, respectively.

The proof follows directly from Definition 6 and Theorem 2.1 of [1], and is omitted. In this paper, we give an $O(mn)$ algorithm for reconstructing a gt-network with m reticulation nodes from a pair of trees T_1 and T_2, each on n leaves.

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-labelled by S. However, events such as hybrid speciation and horizontal gene transfer require non-tree 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 [4, 13, 17]. 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 Figure 2(a), but genes evolve down trees contained in the network as shown in Figures 2(b) and (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 Figure 2(d). 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 Figure 2(e), while all others are inherited from the parent, as in Figure 2(f).

3.1 Maddison's approach to phylogeny reconstruction

In 1997, Wayne Maddison [10] 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. Furthermore, the two trees are related to each other by a single rSPR move, and given the two trees, it is straightforward to construct the network that contained both trees. Maddison also suggested that when the network contains m reticulations, then any two trees contained in the network would be related to each other by a sequence of at most m rSPR moves (though the specific technique for producing the network from two of its constituent trees was not provided). 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. However, finding such a network requires computing the rSPR distance between the two binary gene trees. This is a problem of unknown computational complexity, 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, since any two gene trees may not involve both parents in each reticulation, the best

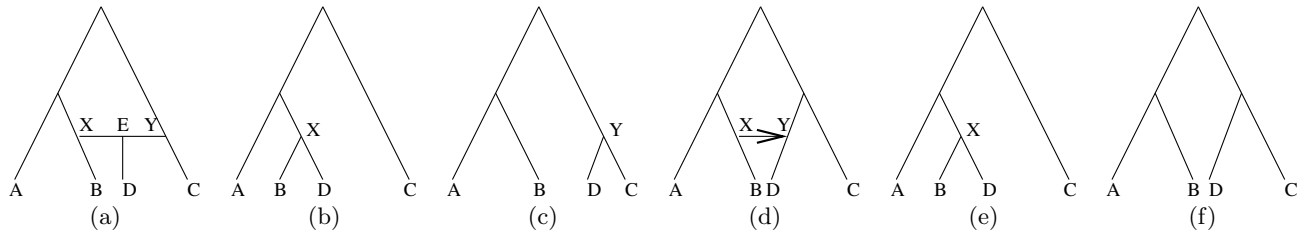

Figure 2: Hybrid speciation: the species network in (a) and its two induced (gene) trees in (b) and (c). Lateral gene transfer: the species network in (d) and its two induced (gene) trees in (e) and (f).

we can hope for is to reconstruct the minimal network that contains both trees; this is what we construct. We address the second limitation in the next section.

We begin by characterizing networks in general, using the model of [9], but with the added constraints that 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, there is at least one tree node (whose two children are also tree nodes) on the directed path between any two reticulation nodes, and if one of the two children of a tree node is a reticulation node, then the other child is a tree node (see [9] for a discussion of the ramifications of missing taxa on reconstructing networks). In this case, we can obtain a nice characterization about the set of gene trees induced by a species network with m reticulations.

THEOREM 2. *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 0 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 Figure 2 for example). 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. Therefore, we have 2^{m+1} distinct trees. □

As the example in Figure 1 illustrated, the gene trees induced by a species network are related through the rSPR operation. We extend this to the general case (with more than one reticulation) in the following theorem.

THEOREM 3. *Let $\mathcal{T}$ be a set of 2^m distinct trees, $T_1, T_2, \ldots, T_{2^m}$. Then, $\mathcal{T}$ is the set of trees induced by a network N with m reticulations if and only if for every tree $T_i \in \mathcal{T}$, $|\{T_j : d_{rSPR}(T_i, T_j) = k\}| = \binom{m}{k}$, where $0 \leq k \leq m$.*

PROOF. We first prove the direction "only if". Let $\mathcal{T}$ be a set of 2^m distinct trees induced by a network with m reticulations; we prove that property in the theorem holds by induction on m.

For the base case of $m = 1$, let $\mathcal{T} = \{T_1, T_2\}$ be the set of two trees induced by a networks N with one reticulation. Let x be the reticulation node in network N, and its two parents are u and v. Assume, without loss of generality, that x is a child of u in T_1, and x is a child of v in T_2. Hence, the the two trees T_1 and T_2 differ only by the location of the subtree rooted at x. Obviously, tree T_2 can be obtained from T_1 by one rSPR move in which the subtree rooted at x is cut from its parent u, and connected to v (see Figure 2 for an example of this scenario). Hence, $d_{rSPR}(T_1, T_2) = 1$.

Assume that every network with m reticulations satisfies the property in the theorem and let N be a network with $m+1$ reticulations, and let x be a reticulation node below which there are no other reticulation nodes. Further, assume u_1 and u_2 are the two parents of x. Let N' be a network with m reticulations obtained from N by removing the subtree T_x rooted at x, along with the two edges (u_1, x) and (u_2, x). Let T be a tree induced by N'. Then, in the set of trees induced by N', there are $\binom{m}{k}$ trees whose rSPR distance from T is k, and $\binom{m}{k-1}$ trees whose rSPR distance from T is $k-1$ (by the induction hypothesis). Each one of those trees will contribute two trees to the set $\mathcal{U}$ of trees induced by N, with the location of T_x being the only difference between the two copies of each tree. Half of those trees will maintain their rSPR distance from T and the other half will increase their rSPR distance (depends on where the subtree T_x is attached in each tree). Therefore, in the set of trees induced by N, the number of trees whose rSPR distance from T is k equals $\binom{m}{k} + \binom{m}{k-1}$, which equals $\binom{m+1}{k}$.

For the "if" part, let $\mathcal{T}$ be a set of 2^m trees that satisfies the property in the theorem. Let T_1 and T_2 be two trees in $\mathcal{T}$ such that $d_{rSPR}(T_1, T_2) = m$. Then, T_2 can be obtained from T_1 by m rSPR operations, where in each operation an edge e_i is cut, thus pruning subtree t_i, which is then regrafted to edge e'_i, for $1 \leq i \leq m$; let $\mathcal{U} = \{t_i : 1 \leq i \leq m\}$. Every tree $T' \in \mathcal{T}$ such that $d_{rSPR}(T', T_1) = q$, $1 \leq q \leq m$, can be obtained from T_1 by q rSPR operations that involve only subtrees from the set $\mathcal{U}$; otherwise, there would exist a tree $T'' \in \mathcal{T}$ such that $d_{rSPR}(T'', T_1) > m$, which contradicts the property in the theorem. The network N is obtained from T_1 by adding a set $\mathcal{E} = \{e^*_i\}$ of edges to tree T_1, where edge e^*_i connects edges e_i and e'_i (directed from e_i to e'_i). The resulting network N has m reticulations,

$u_1, u_2, \ldots, u_m$, and it induces the 2^m trees in $\mathcal{T}$. □

4.1 Efficient reconstruction of gt-networks from gene trees

Theorem 3 implies that given the "full" set of trees induced by a gt-network, we can reconstruct the network via a series of rSPR distance computations among the trees. However, given a pair of trees induced by a gt-network, we can only 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 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 (symbolized by $U(T_1, T_2)$ and $U(T_2, T_1)$ in the proofs). If the two trees are m rSPR moves apart, the marked edges in tree T_1 would form m node-disjoint paths in T_1, and similarly for tree T_2. 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 rSPR move.

LEMMA 1. *Let T_1 and T_2 be two trees induced by a gt-network N. Further, assume that gall Q_x^w in N was broken in the two different ways to obtain T_1 and T_2. 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 $RP^Q(T_2) \subseteq U(T_2, T_1)$. □

LEMMA 2. *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 $RP^Q(T_2) \cap U(T_2, T_1) = \emptyset$. □

Let T be a tree induced by a gt-network N. We denote by $NG(T)$ 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 3. *Let T_1 and T_2 be two trees induced by a 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 $NG(T_2) \cap U(T_2, T_1) = \emptyset$. □

THEOREM 4. *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 in a single way. 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 2, and Lemma 3, and is omitted. Stated differently, Theorem 4 implies that if T_1 and T_2 are two trees induced by a gt-network N such that $d_{rSPR}(T_1, T_2) = m$, then each of the two sets $U(T_1, T_2)$ and $U(T_2, T_1)$ forms m node-disjoint undirected paths in T_1 and T_2, respectively.

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 $END(p_2) = (U_2, V_2)$. 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 5. *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, $d_{rSPR}(T_1, T_2) = m$ 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, $d_{rSPR}(T_1, T_2) = m$. □

THEOREM 6. *Let T_1 and T_2 be two binary trees induced by a gt-network N. We can decide whether $d_{rSPR}(T_1, T_2) = m$ in $O(mn)$ time.*

PROOF. Preprocess the trees so that (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 [3, 7], and (2) For any internal nodes, i, the number $\beta(i)$ of leaves below i can be found in constant time. Further, if S_i is the set of leaves under i, then $LCA(S_i)$ can be found in constant time. This can be achieved in $O(n)$ time using the techniques from [3]. 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 $\beta(i) \neq \beta(LCA(S_i))$ (i is the number assigned to node v). It takes $O(n)$ time to check if $U(T_1, T_2)$ forms a simple path. Further, it takes $O(n)$ time to check if the conditions of Theorem 4 and Theorem 5 hold (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 $d_{rSPR}(T_1, T_2) = m$ in $O(mn)$ time. □

Now, it is straightforward to construct a gt-network N with m galls in $O(mn)$ time, given two induced trees T_1 and T_2 such that $d_{rSPR}(T_1, T_2) = m$.

THEOREM 7. *Let T_1 and T_2 be two binary trees such that $d_{rSPR}(T_1, T_2) = m$. We can decide whether T_1 and T_2 are induced by a gt-network N with m reticulation events, and if so construct such N, in $O(mn)$ time.*

PROOF. By Theorem 6, 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 a number of trees is 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 [14]. 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 binary trees T_1 and T_2 that refine t and such that $d_{rSPR}(T_1, T_2) = 1$.*

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 $d_{rSPR}(T_1, T_2) = 1$. □

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 $d_{rSPR}(T_1, T_2) = 1$ if and only if t_1 and t_2 have a common refinement t that is not fully resolved. Furthermore, we can determine if 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 $d_{rSPR}(T_1, T_2) = 1$. 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 $d_{rSPR}(T_1, T_2) = 1$, 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 4, it follows that $U(T_1, T_2)$ is not a simple path in T_1, and hence $d_{rSPR}(T_1, T_2) \neq 1$; 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 $d_{rSPR}(T_1, T_2) = 1$, 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., $d_{rSPR}(T_1, T_2) = 1$. Let T_1 and T_2 be two binary trees that resolve the two incompatible unresolved trees t_1 and t_2, such that $d_{rSPR}(T_1, T_2) = 1$. By Lemma 6, $p_1 \subseteq U(T_1, T_2)$ and $p_2 \subseteq U(T_2, T_1)$. By Theorem 5, 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 8. *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 $d_{rSPR}(T_1, T_2) = 1$, 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. We preprocess the trees so that: (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 [3, 7], and (2) For any internal node, i, the number $\beta(i)$ of leaves below i can be found in constant time. Further, if S_i is the set of leaves under i, then $LCA(S_i)$ can be found in constant time. This can be achieved in $O(n)$ time using the techniques from [3]. After this preprocessing, computing $U(t_1, t_2)$ takes $O(n)$ time ($O(1)$ time for each edge, and there are $O(n)$ edges). It takes $O(n)$ time to check if $U(t_1, t_2)$ forms a simple path. By Lemmas 7 and 8, we first check whether $U(t_1, t_2)$ and $U(t_2, t_1)$ form simple paths. 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 $d_{rSPR}(T_1, T_2) = 1$. 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 using in $O(n)$ time using Theorem 7. 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, and
- **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".)

6.1 Experimental results

We have done extensive studies evaluating the performance of the SpNet method in simulation, and compared the method to NeighborNet, NJ, and Maddison's method. Not surprisingly, the method outperforms Maddison's method since it is designed to allow for error in the data. Interestingly, it also outperforms NJ when the data are generated on a network with a reticulation (it has essentially identical performance to NJ when the data are generated by a tree, which is not surprising). The comparison between SpNet and NeighborNet is more interesting, and is the focus of our brief discussion here.

We focus here on the results of our experiments on 20-taxon trees and networks with one reticulation. We simulated evolution down these trees and networks (using the tools in [11]) under the GTR model with gamma distributed rates across sites, and invariant sites (using the settings of [18]); reticulations in the networks modelled hybrid speciation. We produced two sequence datasets for each network, one for each of the gene trees contained within the network. The concatenated sequences were given to both NJ and NeighborNet (since those two methods are based on a combined analysis approach), but the separate gene sequences were given to SpNet (since this method is based on a separate analysis approach). Furthermore, concatenated sequences are required by both NJ and NeighborNet because they operate on distance matrices produced from all of the data used in an analysis.

We measured the topological accuracy of the inferred phylogenies (both trees and networks) with respect to the model network as follows. We define $C(N)$–the bipartitions of a network N–as the set of all bipartitions induced by the trees contained inside N; in other words,

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

where $\mathcal{T}(N)$ is the set of all trees induced by network N. Given two networks, N_m (the model network) and N_i (the inferred network), we define the *false positives*, which is the number of incorrectly inferred bipartitions, as $|C(N_i) - C(N_m)|$, and the *false negatives*, which is the number of missing bipartitions, as $|C(N_m) - C(N_i)|$. To obtain the false positive rate (FP) and false negative rate (FN), we normalize both the false positives and false negatives by the number of bipartitions in the model network ($|C(N_m)|$). Hence, the FN rate of any method is at most 100%, whereas the FN rate may be larger than 100%.

False negative and false positive rates below 10% are good, with rates below 5% very good, in evaluating tree reconstruction methods.

Figure 3 shows that all three methods (NeighborNet, SpNet, and NJ) can be distinguished primarily on the basis of their false positive rates. That is, all three 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, NeighborNet has very poor false positive rates on both trees and networks with one reticulation, even at very long sequences (4000 nucleotides). In particular, our method, SpNet, has an excellent (i.e., very low) false positive rate, but NeighborNet has a very high false positive rate, even at long sequences. Furthermore, NeighborNet's false negative rate is quite comparable to that of SpNet, and so the huge difference in false positive rates is very important.

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.

7. CONCLUSIONS AND ACKNOWLEDGMENTS

Our experiments show that NeighborNet, the current best 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) non-binary 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

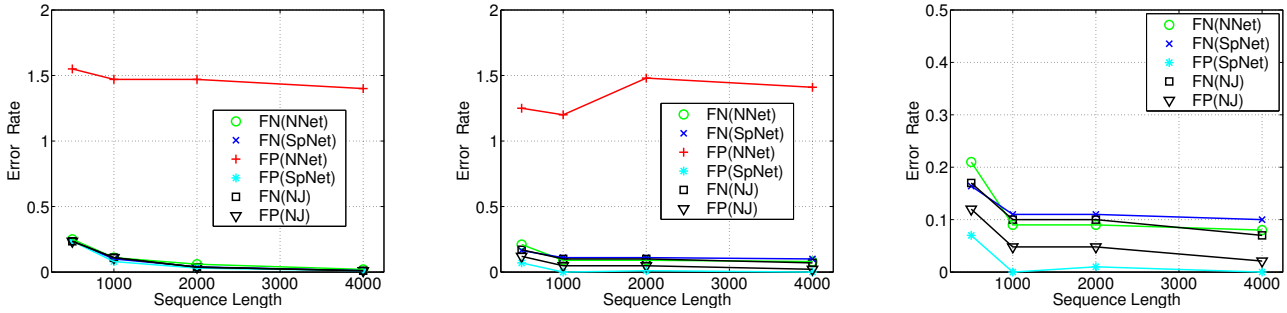

Figure 3: FN and FP error rates of NeighborNet (NNet) and SpNet on 20-taxon networks, with 0.1 scaling factor, with tree model phylogeny (left), and 1-hybrid network (middle). The rightmost graph shows the results without the FP rate of NNet.

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.

This work is supported by National Science Foundation under grants DEB 01-20709 (Linder & Warnow), EIA 01-21651 (Warnow), EIA 01-13654 (Warnow), EIA 01-21680 (Linder & Warnow), EF 01-31453 (Linder & 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).

8. REFERENCES

[1] B. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–13, 2001.

[2] D. Bryant and V. Moulton. NeighborNet: An agglomerative method for the construction of planar phylogenetic networks. In R. Guigo and D. Gusfield, editors, *Proc. 2nd Workshop Algorithms in Bioinformatics (WABI'02)*, volume 2452 of *Lecture Notes in Computer Science*, pages 375–391. Springer Verlag, 2002.

[3] W.H.E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 2:7–28, 1985.

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

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

[6] D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proceedings of Computational Systems Bioinformatics (CSB 03)*, 2003.

[7] D. Harel and R.E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

[8] J.G. Lawrence and H. Ochman. Reconciling the many faces of lateral gene transfer. *Trends in Microbiology*, 10:1–4, 2002.

[9] C.R. Linder, B.M.E. Moret, L. Nakhleh, A. Padolina, J. Sun, A. Tholse, R. Timme, and T. Warnow. Phylogenetic networks: generation, comparison, and reconstruction. Technical Report TR-CS-2003-26, University of New Mexico, 2003.

[10] W.P. Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.

[11] L. Nakhleh, J. Sun, T. Warnow, C.R. Linder, B.M.E. Moret, and A. Tholse. Towards the development of computational tools for evaluating phylogenetic network reconstruction method. In *Proc. 8th Pacific Symposium on Biocomputing (PSB 03)*, pages 315–326, 2003.

[12] R. Page and M.A. Charleston. Trees within trees: Phylogeny and historical associations. *Trends in Ecology and Evolution*, 13:356–359, 1998.

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

[14] H. Shimodaira and M. Hasegawa. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular Biology and Evolution*, 16:1114–1116, 1999.

[15] L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.

[16] T. Warnow. Tree compatibility and inferring evolutionary history. *Journal of Algorithms*, 16:388–407, 1994.

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

[18] D. Zwickl and D. Hillis. Increased taxon sampling greatly reduces phylogenetic error. *Systematic Biology*, 51(4):588–598, 2002.