Phylogenetic Networks, Trees, and Clusters

Luay Nakhleh[1] and Li-San Wang[2]

[1] Department of Computer Science
Rice University
Houston, TX 77005, USA
nakhleh@cs.rice.edu
[2] Department of Biology
University of Pennsylvania
Philadelphia, PA 19104, USA
lswang@mail.med.upenn.edu

Abstract. Phylogenetic networks model evolutionary histories in the presence of non-treelike events such as hybrid speciation and horizontal gene transfer. In spite of their widely acknowledged importance, very little is known about phylogenetic networks, which have so far been studied mostly for specific datasets.
Even when the evolutionary history of a set of species is non-treelike, individual genes in these species usually evolve in a treelike fashion. An important question, then, is whether a gene tree is "contained" inside a species network. This information is used to detect the presence of events such as horizontal gene transfer and hybrid speciation. Another question of interest for biologists is whether a group of taxa forms a clade based on a given phylogeny. This can be efficiently answered when the phylogeny is a tree simply by inspecting the edges of the tree, whereas no efficient solution currently exists for the problem when the phylogeny is a network. In this paper, we give polynomial-time algorithms for answering the above two questions.

1 Introduction

Phylogenies are the main tool for representing the relationships among biological entities. Their pervasiveness has led biologists, mathematicians, and computer scientists to design a variety of methods for their reconstruction. Furthermore, extensive studies have been focused on the performance of these methods under different models and settings, as well as on the combinatorial and biological properties of trees (e.g., [7, 2]). However, almost all such methods construct trees, and almost all studies have been aimed at trees. Yet, biologists have long recognized that trees oversimplify our view of evolution, since they cannot take into account such events as hybridization, lateral gene transfer, and recombination. These non-tree events give rise to edges that connect nodes on different branches of a tree, giving rise to a directed acyclic graph structure that is usually called a *phylogenetic network*.

A gene tree is a model of how a gene evolves through duplication, loss, and nucleotide substitution. Gene trees can differ from one another as well as from the species phylogeny. Such differences arise during the evolutionary process due to events such as duplication and loss, whereby each genome may end up with multiple copies of a given

gene—but not necessarily the same copies that survive in another genome. Unless the genome is very well sampled, only a subset (sometimes only one copy, in fact) of the gene is used in phylogenetic analyses. As a result, the phylogeny for the gene may not agree with the species phylogeny, nor with the phylogeny for another gene. Because the gene copy has a single ancestral copy, barring recombination, the resulting history is a branching tree. Point mutations can cause some of the copies to be imperfect representations of the original, but this process does not compromise the existence of the (gene) tree. Events such as recombination, hybrid speciation, and lateral gene transfer break up the genomic history into many small pieces, each of which has a strictly tree-like pattern of descent [4]. Thus, within a species phylogeny, many tangled gene trees can be found, one for each nonrecombined locus in the genome. Incongruence among gene trees is a powerful tool for detecting recombination, hybrid speciation, and other non-treelike evolutionary events (e.g., see [6]). While testing for incongruence between two (gene) trees can be done in a straightforward manner, it is not as simple for testing the incongruence between a tree and a network, since the number of trees "inside" a network grows exponentially with the number of non-treelike events. In this paper, we give the first polynomial-time algorithm for solving this problem.

A phylogeny can be viewed as a collection of clusters of taxa (each defined as the set of leaves in a subtree). Various approaches for reconstructing phylogenies (trees and networks) have been proposed based on this view (see, e.g., [1, 3]). An interesting biological question, then, is whether a group of taxa forms a cluster in a given phylogeny. This question can be answered in a straightforward manner when the phylogeny is a tree, since each edge in a tree defines a unique cluster. However, the number of clusters in a phylogenetic network grows exponentially with the number of non-treelike events, and hence an efficient algorithm for solving the problem is not straightforward. In this paper, we present the first polynomial-time algorithm for solving this problem.

2 Phylogenetic Trees and Networks

A *rooted* phylogenetic tree is a rooted tree without redundant nodes (nodes of indegree and outdegree 1) and whose leaves are labelled distinctively (by a set of taxa). An unrooted phylogenetic tree is a rooted phylogenetic tree with the root suppressed. Every edge e in an unrooted leaf-labeled tree T defines a bipartition (or, split) $\pi(e)$ on the leaves (induced by the deletion of e), so that we can define the set $\Pi(T) = \{\pi(e): e \in E(T)\}$. Every edge e in a rooted leaf-labeled tree T defines a cluster $c(e)$ of leaves (those leaves that are reachable from the root through e), so that we can define the set $C(T) = \{c(e): e \in E(T)\}$. A *clade* of a rooted tree T is the entire subtree rooted at a node of T; the set of all leaves in a clade correspond to a *cluster* of T.

Evolutionary events such as hybrid speciation and lateral gene transfer result in networks of relationships. In hybrid speciation, two lineages recombine to create a new species, as symbolized in Fig. 1(a). Prior to hybridization, each site on each homolog has evolved in a tree-like fashion, although, due to meiotic recombination, different strings of sites may have different histories. Thus, each site in the homologs of the parents of the hybrid evolved in a tree-like fashion on one of the trees contained inside

(or induced by) the network representing the hybridization event, as illustrated in Fig. 1(b) and Fig. 1(c).

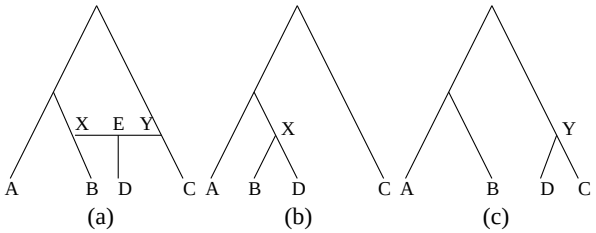

Fig. 1. A network with a single hybrid speciation event, and its two induced trees.

2.1 Phylogenetic Networks: Model and Properties

We adopt the general model of (reduced) phylogenetic networks given in [5].

Definition 1. *A* **phylogenetic network** *is a connected directed acyclic graph* $N = (V, E)$*, where* V *can be partitioned into* $\{r\} \cup Tr(N) \cup Nt(N) \cup L(N)$*, where:*

1. *Node* r *is the root; it has indegree* 0*.*
2. *Set* $Tr(N)$ *is the set of tree nodes; each node* u *in* $Tr(N)$ *has indegree* 1 *and outdegree* > 1*.*
3. *Set* $Nt(N)$ *is the set of network nodes; each node* v *in* $Nt(N)$ *has indegree* 2 *and outdegree* 1*.*
4. *Set* $L(N)$ *is the set of leaf nodes (taxa); each node* x *in* $L(N)$ *has indegree* 1 *and outdegree* 0*. Each node* x *in* $L(N)$ *is labeled uniquely by an integer* i*, where* $1 \leq i \leq |L(N)|$*.*

Figure 1(a) shows an example of a phylogenetic network. Given a network N, we classify its edges as *tree edges* and *network edges*. An edge $e = (u, v)$ is a tree edge if v is a tree node or a leaf; otherwise, it is a network edge. We say that network N is *binary* if the root and all tree nodes of N have outdegree 2. In this paper, and unless noted otherwise, all networks are binary. Further, we assume that if u is a tree node and (u, v) and (u, w) are the two edges incident from u, then at least one of the two nodes v and w is a tree node.

A *forced contraction* is an operation on a graph in which we delete a redundant node and replace the two edges incident to it by a single edge. We generalize the *clade* concept to networks as follows. Given a network N, we say that the DAG N', rooted at node x, is a *network clade* of N, if there exists an edge $e = (u, x)$ in N whose removal disconnects N, thus creating two components, one of which is N' (rooted at x). If network clade N' does not contain network nodes, i.e., N' is a tree, we refer to

N' simply as a clade. Given a network N, and a clade N', we say that N' is *maximal* if N does not contain any clade N'' such that $N' \subset N''$.

If there is a directed path from node u to node v ($u \neq v$) in a directed tree T, we say that u is *above* v and that v is *below* u, both denoted by $u > v$. We say that node u in N is a *lowest network node* if (1) u is a network node, and (2) for any network node v, $v \neq u$, we have $u \not\rightarrow v$. We write $N|_{L'}$, where $L' \subset L(N)$, to denote the subgraph N' obtained from N by removing all leaves not in L', and then applying forced contraction operations and removal of nodes of outdegree 0 (other than the leaves in L').

2.2 The Relationship Between Networks and Trees

There is a fundamental biological relationship between (species) networks and (gene) trees, as described above. We now formalize this concept mathematically.

Let N be a network with p network nodes $h_1, h_2, \ldots, h_p$. Further, assume that the two edges incident into h_i are e_{i_1} and e_{i_2}. An *inheritance profile*, IP, for N is a set of size p which contains exactly one of the two edges e_{i_1} and e_{i_2} for each network nodes h_i. A rooted tree T is *induced* by (or, contained in) a network N if there exists an inheritance profile IP such that T can be obtained from N as follows: for network node h_i, if $e_{i_1} \in IP$, remove edge e_{i_2}; otherwise, remove edge e_{i_1} (and then apply forced contraction operations to the resultant graph). Biologically, the evolutionary history of a gene within the species network corresponds to a tree T induced by N. Associated with this tree is an inheritance profile IP that determines how to obtain T from N; in this case, we say that IP is a *valid* inheritance profile that induces T. We denote by $\mathcal{T}(N)$ the set of all trees induced by network N. A network N induces (or, contains) a cluster C, $C \subseteq L(N)$, if there exists a tree T such that N induces T and C is a cluster of T. We denote by $\mathcal{C}(N)$ the set of all clusters induced by network N.

As mentioned before, deciding whether a cluster or a tree are induced by a given network plays a significant role in solving major biological problems such as network reconstruction, gene tree and species network relationships. We now formalize the two decision problems.

Problem 1. (THE NETWORK-TREE CONTAINMENT PROBLEM)

Input: A phylogenetic network N and a tree T.
Question: Does N contain T?

Problem 2. (THE NETWORK-CLUSTER CONTAINMENT PROBLEM)

Input: A phylogenetic network N and a cluster C.
Question: Does N contain C?

A trivial approach for solving the Network-Cluster Containment Problem is to find "the" lowest common ancestor, x, of C in the network N, and test whether the cluster is contained in the network clade rooted at x. This approach may fail for (at least) two reasons: (1) x may not be unique in a network, and (2) the network clade rooted at x may contain many of the network nodes of N, in which case the search for a solution would take time that is exponential in the number of network nodes, and hence, probably the network size. In Section 5 we give polynomial time algorithms for these two problems. In order to obtain these results, we first introduce the concept of *network decomposition* which forms the basis for our algorithms.

3 Network Decomposition

Before we give the technical details of our algorithms, we describe the network representation we use, which is vital for achieving the running times of the algorithms in the next sections. We assume that a network N is represented using an $n \times n$ adjacency matrix M^N, where n is the number of nodes in the network. We have $M^N[u, v] = 1$ if there is an edge $(u, v) \in E(N)$, and $M^N[u, v] = 0$ otherwise. Using this representation, a forced contraction operation takes $O(1)$ time, and an edge deletion takes $O(1)$ time, as well.

3.1 Preprocessing Networks

An *SH-loop* (speciation-hybridization) is a cycle that contains only network edges, and that consists of two paths p_1 and p_2, such that p_1 and p_2 starting from the same tree node v_0, pass through two sets of network nodes, and end at the same network node v_1. Let $e_1 = (v_0, x)$ and $e_2 = (v_0, y)$ be the two network edges incident from v_0. We break the SH-loop by removing either e_1 or e_2, and applying forced contraction operations to all redundant nodes. We repeat the same process until N is SH-loop-free, i.e., N does not contain any SH-loops.

Theorem 1. *A network $N = (V, E)$ can be preprocessed in $O(|E(N)|^2)$ time.*

Proposition 1. *Let N be a phylogenetic network, and let N' be the network obtained after the preprocessing. Then, $\mathcal{T}(N) = \mathcal{T}(N')$.*

3.2 Maximal Clades and Connections

Unless noted otherwise, all networks are SH-loop free. Given a phylogenetic network N, we seek to decompose N into maximal-size clades and disjoint subgraphs of N that connect those clades. To formalize this, we first define some concepts.

Given a node x in network N, we say that a network node y ($y \neq x$) in N is *x-convergent* if any directed path from y to a leaf of N passes through x. Given a maximal clade A of N, and the root a of A, we say that subgraph J of N is the *connection* of A if J is the subgraph obtained by restricting N to all a-convergent nodes and their incident edges.

Lemma 1. *Let A and J be a clade and its connection, respectively, in a network N. Then, when reversing the orientation of its edges, J has a rooted tree topology, where each leaf is a tree node in N and each internal node is a network node in N. Further, the root of J is a lowest network node.*

Definition 2. *A* T-decomposition *of a network N is an ordered set of pairs $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$, where A_i and J_i are a maximal clade and its connection, respectively, in N_i (N_i is obtained by removing the subgraphs A_{i-1} and J_{i-1} from N_{i-1}, except for the leaves of J_{i-1}, i.e., the tree nodes, and applying forced contraction operations to the resultant graph; for the base case, $N_1 = N$); m is the cardinality of the decomposition.*

Theorem 2. *A T-decomposition of a network N is computable in $O(|Nt(N)||V(N)|)$ time.*

Let (u, v) be a terminal edge (i.e., and edge incident with a leaf) that belongs to connection J_i; v is a tree node in N. If N is binary, then for the three edges incident to v, two belong to the same component, because v is suppressed in the i^{th} step in the decomposition algorithm. We define $\iota(u, v)$ to be the index of this component. It is straightforward to show that $\iota(u, v) > i$. Finally, we show that exactly one terminal edge from each component in a T-decomposition is used to induce a tree T.

Lemma 2. *If T is a tree induced by a network N, and D is a T-decomposition of N, then exactly one terminal edge from each connection in D is used to induce that tree.*

3.3 Dependency Graphs

Given a network N and its T-decomposition D, we define the *dependency digraph* $K^{N,D}$ as follows.

Definition 3. *Given a network N and its T-decomposition $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$, the dependency graph is a directed multigraph $K^{N,D}$, where node v_i in $K^{N,D}$ corresponds to the pair (A_i, J_i) in D, and edge (v_i, v_j) $(i > j)$ in $K^{N,D}$ corresponds to a terminal edge connecting J_j and J_i in N.*

In other words, $K^{N,D}$ is the graph resulting from replacing each component (A_i, J_i) in D by a single node v_i, and hence, $K^{N,D}$ is necessarily connected. If $K^{N,D}$ had a cycle, then N would be cyclic. Therefore, we have the following result.

Proposition 2. *The dependency graph $K^{N,D}$ is connected and acyclic, and is computable in $O(|Nt(N)||V(N)|)$ time. Moreover, (v_i, v_j) is an edge in $K^{N,D}$ only if $i > j$.*

4 Reduced Inheritance Profiles and the Cluster Lemma

Given a T-decomposition D of cardinality m, a *reduced inheritance profile* is a set of size m that contains exactly one terminal edge per connection in the decomposition. We only keep the terminal edges because all inheritance profiles having the same set of terminal edges necessarily induce the same tree. A reduced inheritance profile extends into an inheritance profile in a straightforward manner, as no edges in the reduced inheritance profile are incident with the same network node. We say that a reduced inheritance profile is *valid* if it induces a tree. The following results show the correspondence between inheritance profiles and reduced inheritance profiles.

Proposition 3. *Let D be a T-decomposition of a network N. Then,*

1. *For each valid reduced inheritance profile IP there exists a valid inheritance profile IP' that contains IP and induces the same tree.*
2. *For each valid inheritance profile IP' there exists a unique valid reduced inheritance profile IP that induces the same tree.*

The dependency graph can be seen as a compact representation, mainly for reduced inheritance profiles.

Lemma 3. *Let D be a T-decomposition of a network N, $K^{N,D}$ be the dependency graph, and IP be a valid reduced inheritance profile. Then, $K^{N,D}$, restricted to the edges in IP, forms a tree.*

We are now in position to show the correlation between clusters and a T-decomposition of a network – a result that forms the basis for our algorithms.

Lemma 4. (Cluster Lemma) *Let $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$ be a T-decomposition of a network N. Each cluster C induced by N can be written as $C = \cup_j C_j$, where each C_j is an element of $\{L(A_i) : 1 \leq i \leq m\}$, except for at most one of the C_j's, which may be a proper subset of an element of $\{L(A_i) : 1 \leq i \leq m\}$.*

Corollary 1. *Let $D = \{(A_i, J_i)\}_{1 \leq i \leq m}$ be a T-decomposition of a network N, IP be a reduced inheritance profile, and C be a cluster. Then, when restricted to the nodes whose corresponding maximal clades have nonempty intersection with C and to the edges in IP, the dependency graph $K^{N,D}$ forms a tree. Further, the root of that tree has the highest index.*

Corollary 1 gives an algorithm for computing the component that contains the node which "determines" a cluster C.

5 Polynomial-time Algorithms for the Decision Problems

5.1 Deciding the Network-Cluster Containment Problem

We are finally in a position to describe a polynomial-time algorithm for deciding the Network-Cluster Containment Problem. The algorithm is given in Fig. 2. Let $D = \{A_i, J_i\}_{1 \leq i \leq m}$ be a T-decomposition of a network N, and let $C \subset L(N)$ be a cluster. We define the set $\psi(C) = \{i : 1 \leq i \leq m$ and $L(A_i) \subseteq C\}$. The basic idea is to compute a set E_C of edges that are *incompatible* with C, i.e., edges that cannot co-exist with C in the same tree induced by N.

Theorem 3. *Algorithm TestCinN(N,C) decides the Network-Cluster Containment Problem in $O(|V(N)|^2)$ time.*

Corollary 2. *Given any network N, a phylogenetic tree T, and a cluster C in T, N induces T if and only if N_C induces T.*

5.2 Deciding the Network-Tree Containment Problem

Using algorithm TestCinN(N,C), Fig. 3 describes our polynomial-time algorithm for deciding the Network-Tree Containment Problem.

Theorem 4. *Algorithm TestTinN(N,T) decides the Network-Tree Containment Problem in $O(|V(N)||L(N)|)$ time.*

Algorithm TestCinN(N,C)

1. Compute a T-decomposition $D = ((A_1, J_1), \ldots, (A_m, J_m = \emptyset))$.
2. Test if C can be decomposed into the following form: $\bigcup_{i \in \psi(C)} L(A_i) \cup L'$, where $L' = \emptyset$ or $L' \subset L(A_l)$ for some l. If not, return NO. If $L' = \emptyset$ then let $l = \max_{i \in \psi(C)} i$.
3. Partition $V = V(K^{N,D})$ into two sets: $V_C = \{v_i | i \in \psi(C)\}$ and $\overline{V_C} = V - V_C$. Compute the set $E_C = \{e_{ij} = (v_i, v_j) | e_{ij} \in E(K^{N,D}), v_i \in \overline{V_C}, v_j \in V_C, j \neq l\}$.
4. If $L' \neq \emptyset$, test if L' is a cluster of A_l. If not, return NO; otherwise:
   (a) Let v' be the root of the clade whose leaf set is L'.
   (b) For each terminal edge (u, v) in A_i, for some $i \in \psi(C)$ and $\iota(u, v) = l$, (edge (u, v) connects the i'th component to the l'th component), add (u, v) to E_C if u is not a descendant of v' in N_i.
5. Remove all terminal edges in N and that correspond to edges in E_C (and apply forced contraction operations); let the result be N_C. If N_C is connected, return YES. Otherwise, return NO.

Fig. 2. Algorithm **TestCinN** for deciding the NETWORK-CLUSTER CONTAINMENT PROBLEM.

Algorithm TestTinN(N,T)

1. Compute a T-decomposition $D = ((A_1, J_1), \ldots, (A_m, J_m = \emptyset))$.
2. For each nontrivial cluster C in T ($C \neq L(N)$ and $|C| > 1$), call TestCinN(N,C); update N by removing E_C from N.
3. If N is connected, return YES; otherwise, return NO.

Fig. 3. Algorithm **TestTinN** for deciding the NETWORK-TREE CONTAINMENT PROBLEM.

References

[1] 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.

[2] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Inc., Sunderland, MA, 2003.

[3] D.H. Huson. SplitsTree: A program for analyzing and visualizing evolutionary data. *Bioinformatics*, 14(1):68–73, 1998.

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

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

[6] L. Nakhleh, T. Warnow, and C.R. Linder. Reconstructing reticulate evolution in species – theory and practice. In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2004)*, pages 337–346, 2004.

[7] D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. In D.M. Hillis, B.K. Mable, and C. Moritz, editors, *Molecular Systematics*, pages 407–514. Sinauer Assoc., Sunderland, Mass., 1996.