INFERRING PHYLOGENETIC RELATIONSHIPS AVOIDING FORBIDDEN ROOTED TRIPLETS

YING-JUN HE, TRINH N.D. HUYNH, JESPER JANSSON, AND WING-KIN SUNG*

School of Computing, National University of Singapore,
3 Science Drive 2, Singapore 117543.
E-mail: {heyingju,huynhngo,jansson,ksung}@comp.nus.edu.sg

To construct a phylogenetic tree or phylogenetic network for describing the evolutionary history of a set of species is a well-studied problem in computational biology. One previously proposed method to infer a phylogenetic tree/network for a large set of species is by merging a collection of known smaller phylogenetic trees on overlapping sets of species so that no (or as little as possible) branching information is lost. However, little work has been done so far on inferring a phylogenetic tree/network from a specified set of trees when in addition, certain evolutionary relationships among the species are known to be highly unlikely. In this paper, we consider the problem of constructing a phylogenetic tree/network which is consistent with all of the rooted triplets in a given set $\mathcal{T}$ and none of the rooted triplets in another given set $\mathcal{F}$. Although NP-hard in the general case, we provide some efficient exact and approximation algorithms for a number of biologically meaningful variants of the problem.

1. Introduction

The evolutionary relationships among a set of species S are commonly described by a phylogenetic tree or a phylogenetic network. A *phylogenetic tree* is a rooted, unordered tree whose leaves are distinctly labeled by S and where each internal node represents an ancestral species and each edge represents the evolution from one species to another (see, e.g., [19, 25]). However, scientists have observed that certain evolutionary events cannot be described properly using the treelike model; examples of these so-called *recombination events* include horizontal gene transfer and hybrid speciation [10, 12, 20, 21, 23, 27]. *Phylogenetic networks* were proposed as a way to represent non-treelike evolution by extending the definition of phylogenetic trees to allow every node to have more than one parent.

One approach to constructing large phylogenetic trees/networks is by combining a set of known trees into one supertree/network [3, 4, 11, 12, 17, 18, 21, 22, 24, 26]. In this paper, we focus on the problem of constructing a phylogenetic tree/network from *rooted triplets* (i.e., binary phylogenetic trees with exactly three leaves each). Variants of this problem have been studied previously in [1, 5, 8, 9, 11, 13, 14, 15, 16, 17, 28]. The motivation for the rooted triplets approach is that a highly accurate tree for just three species can be obtained through maximum likelihood-based methods [6] or Sibley-Ahlquist-style DNA-DNA hybridization experiments (see [17]). Moreover, when applying those methods, apart

*Dr. Sung is also affiliated with Genome Institute of Singapore, 60 Biopolis Street, Genome, Singapore 138672.

from obtaining a set of reliable rooted triplets, we may also discover some rooted triplets (referred to as *forbidden rooted triplets*) which are very unlikely to appear as induced subgraphs in the true tree/network. Other than [5, 22], little work has been done to study whether the extra information provided by forbidden rooted triplets can be used in phylogenetic reconstruction. Therefore, in this paper, we investigate some problems related to constructing a phylogenetic tree or phylogenetic network from a given set $\mathcal{T}$ of "good" rooted triplets and a given set $\mathcal{F}$ of forbidden rooted triplets.

1.1. *Problem definitions and summary of our results*

A *phylogenetic tree* is a rooted, unordered tree whose leaves are labeled in such a way that all leaf labels are disjoint, and furthermore, all of its internal nodes have outdegree at least 2. A rooted tree is *binary* if all of its internal nodes have precisely outdegree 2. A binary phylogenetic tree with three leaves is called a *rooted triplet*. The unique rooted triplet on a leaf set $\{x, y, z\}$ in which the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of x and z (or equivalently, where the lowest common ancestor of x and y is a proper descendant of the lowest common ancestor of y and z) is denoted by $(\{x, y\}, z)$. A *binary caterpillar tree* is a rooted binary tree where every internal node has at least one child which is a leaf.

A *phylogenetic network* is a generalization of a binary phylogenetic tree formally defined as a rooted, directed acyclic graph where: (1) exactly one node has indegree 0 (the *root*); (2) all other nodes have indegree 1 or 2; (3) all nodes with indegree 2 (referred as *recombination nodes*) have outdegree 1, and all other nodes have outdegree 0 or 2; and (4) all nodes with outdegree 0 (the *leaves*) are distinctly labeled. For any phylogenetic network N, let $\mathcal{U}(N)$ be the undirected graph obtained from N by replacing each directed edge by an undirected edge. N is said to be a *galled phylogenetic network* (or *galled network*, for short) if all cycles in $\mathcal{U}(N)$ are node-disjoint. Galled networks form an important class of phylogenetic networks (see, e.g., [10] for a discussion) and are also known in the literature as *topologies with independent recombination events* [27], *galled-trees* [10], *gt-networks* [21], and *level*-1 *phylogenetic networks* [7, 16].

Let N be a phylogenetic network. A rooted triplet t is said to be *consistent with* N if t is an induced subgraph of N, and a set $\mathcal{T}$ of rooted triplets is *consistent with* N if every rooted triplet in $\mathcal{T}$ is consistent with N. The set of all rooted triplets which are consistent with N is denoted by $\mathcal{R}(N)$, and we let $N(\mathcal{T})$ be the subset of $\mathcal{T}$ containing all rooted triplets from $\mathcal{T}$ that are consistent with N, i.e., $N(\mathcal{T}) = \mathcal{T} \cap \mathcal{R}(N)$.

Denote the set of leaves in a phylogenetic tree/network N by $\Lambda(N)$, and for any set $\mathcal{T}$ of rooted triplets, define $\Lambda(\mathcal{T}) = \bigcup_{t_i \in \mathcal{T}} \Lambda(t_i)$. Given a leaf set L, a set $\mathcal{T}$ of rooted triplets is called *dense (with respect to L)* if $\Lambda(\mathcal{T}) = L$ and for each $\{x, y, z\} \subseteq L$, at least one of the three possible rooted triplets $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ belongs to $\mathcal{T}$. Finally, for any set $\mathcal{T}$ of rooted triplets and $L' \subseteq \Lambda(\mathcal{T})$, we define $\mathcal{T} \mid L'$ as the subset of $\mathcal{T}$ consisting of all rooted triplets $(\{x, y\}, z)$ with $\{x, y, z\} \subseteq L'$.

Given two sets $\mathcal{T}$ and $\mathcal{F}$ of rooted triplets, we study the following problems. Throughout this paper, we let L represent the leaf set $\Lambda(\mathcal{T}) \cup \Lambda(\mathcal{F})$ and we write $n = |L|$.

- *The mixed triplets problem* (MT): Construct a phylogenetic network N with $\Lambda(N) = L$ such that $\mathcal{T} \subseteq \mathcal{R}(N)$ and $\mathcal{F} \cap \mathcal{R}(N) = \emptyset$, if such an N exists; otherwise, output *null*. In Section 3.1, we show that this problem is NP-hard in its general form. In Section 3.2, we investigate a restricted case of the problem where $\mathcal{F}$ consists of disjoint rooted triplets, i.e., where $\Lambda(t) \cap \Lambda(t') = \emptyset$ for any $t, t' \in \mathcal{F}$ with $t \neq t'$, and show how to solve this case efficiently in $O(n \log n)$ time.
- *The mixed triplets problem restricted to trees* (MTT): Construct a phylogenetic tree T with $\Lambda(T) = L$ such that $\mathcal{T} \subseteq \mathcal{R}(T)$ and $\mathcal{F} \cap \mathcal{R}(T) = \emptyset$, if such a T exists; otherwise, output *null*. Note that T is not required to be a binary phylogenetic tree here. In Section 2.1, we describe an $O(|\mathcal{T}| \cdot n + |\mathcal{F}| \cdot n^2)$-time algorithm for MTT. We also study a corresponding maximization problem that we call MMTT which asks for a phylogenetic tree T that maximizes $|T(\mathcal{T})| - |T(\mathcal{F})|$. MMTT is NP-hard, so we present an algorithm for inferring a phylogenetic tree T that guarantees $|T(\mathcal{T})| - |T(\mathcal{F})| \geq \frac{1}{3} \cdot (|\mathcal{T}| - |\mathcal{F}|)$ in Section 2.2.
- *The mixed triplets problem restricted to galled networks* (MTG): Construct a galled network N with $\Lambda(N) = L$ such that $\mathcal{T} \subseteq \mathcal{R}(N)$ and $\mathcal{F} \cap \mathcal{R}(N) = \emptyset$, if such an N exists; otherwise, output *null*. This problem is NP-hard even if restricted to $\mathcal{F} = \emptyset$ [15]; therefore, MTG for arbitrary $\mathcal{F}$ is also NP-hard. In Section 4.1, we study the maximization version of MTG called MMTG and give an algorithm for inferring a galled network N that guarantees $|N(\mathcal{T})| - |N(\mathcal{F})| \geq \frac{5}{12} \cdot (|\mathcal{T}| - |\mathcal{F}|)$. We also consider the restricted case of MTG where $\mathcal{T}$ is dense, and show that this restricted case can be solved in $O(n^4)$ time in Section 4.2.

Below, the elements in $\mathcal{F}$ are called *forbidden rooted triplets*.

1.2. *Related results*

Several papers have previously studied MT, MTT, MMTT, MTG, MMTG, and some of their variants for the special case $\mathcal{F} = \emptyset$. Aho, Sagiv, Szymanski, and Ullman [1] presented an $O(|\mathcal{T}| \cdot n)$-time algorithm for determining whether a given set $\mathcal{T}$ of rooted triplets on n leaves is consistent with some rooted, distinctly leaf-labeled tree, and if so, returning one (i.e., MTT restricted to $\mathcal{F} = \emptyset$)[a]. Henzinger, King, and Warnow [11] later showed how to implement the algorithm of Aho *et al.* to run asymptotically faster. Gąsieniec, Jansson, Lingas, and Östlin [8] considered a version of the problem where the leaves in the output tree are required to comply with a left-to-right leaf ordering given as part of the input. Related optimization problems where the objective is to construct a rooted tree consistent with the maximum number of rooted triplets in the input (i.e., MMTT with $\mathcal{F} = \emptyset$) or to find a maximum cardinality subset L' of $\Lambda(\mathcal{T})$ such that $\mathcal{T} \mid L'$ is consistent with some tree have been studied in [5, 9, 13, 28] and [14], respectively. We remark that MMTT with $\mathcal{F} = \emptyset$ is NP-hard (see [5], [13], or [28]) and approximable within a factor of $\frac{1}{3}$ in polynomial time [9] (meaning that the approximation algorithm in [9] always outputs a

[a] In contrast, the analog of this problem for *unrooted* trees is NP-hard, even if all of the input trees are *quartets* (unrooted, distinctly leaf-labeled trees each having four leaves and no nodes of degree two) [26]. See also [18].

phylogenetic tree which is consistent with at least $\frac{1}{3} \cdot |\mathcal{T}|$ of the rooted triplets in $\mathcal{T}$).

As for inferring a phylogenetic network from a given set of rooted triplets on n leaves (i.e., MT with $\mathcal{F} = \emptyset$), Jansson and Sung [16] proved that if no restrictions are placed on the structure of the output phylogenetic network then the problem always has a solution which can easily be obtained from any given sorting network for n elements. [16] also presented an $O(n^6)$-time algorithm for inferring a galled network (if one exists) consistent with a given dense set of rooted triplets on n leaves; Jansson, Nguyen, and Sung [15] subsequently reduced its running time to $O(n^3)$, which is optimal since the size of the input is $O(n^3)$ in the dense case. In [15], it was also proved that the problem becomes NP-hard for non-dense inputs (i.e., MTG with $\mathcal{F} = \emptyset$), and that the corresponding optimization problem (MMTG with $\mathcal{F} = \emptyset$) is approximable within a factor of $\frac{5}{12}$ in polynomial time.

Also in the context of inferring a phylogenetic network from a set of trees, Nakhleh, Warnow, and Linder [21] gave an algorithm for inferring a galled network from two binary phylogenetic trees with identical leaf sets. In addition, they studied the case where the input trees may contain errors but where only one recombination node is allowed in the network. Huson, Dezulian, Klöpper, and Steel [12] addressed a similar problem for constructing an *unrooted* phylogenetic network from a set of unrooted, distinctly leaf-labeled trees.

For the general case $\mathcal{F} \neq \emptyset$, much less is known. Bryant [5] showed that MTT restricted to $\mathcal{T} = \emptyset$ is NP-hard if the output is required to be a binary tree, but solvable in polynomial time if we further restrict the solution to be a binary caterpillar tree. However, given a set $\mathcal{S}$ of binary phylogenetic caterpillar trees whose leaf sets are subsets of a label set L, it is NP-hard to determine if there exists a binary tree T with $\Lambda(T) = L$ such that no tree in $\mathcal{S}$ is an induced subgraph of T, even if T is restricted to be a binary caterpillar tree [22].

2. Algorithms for MTT and MMTT

2.1. *A polynomial-time algorithm for MTT*

Here, we present a polynomial-time algorithm for solving MTT. It is a generalization of the algorithm of Aho *et al.* [1] for determining if a given set $\mathcal{T}$ of rooted triplets is consistent with a rooted tree and if so, constructing one. We extend their algorithm to deal with a nonempty set $\mathcal{F}$ of forbidden rooted triplets. Note that if the output tree is constrained to be binary then the problem becomes NP-hard even if $\mathcal{T} = \emptyset$ (see Section 1.2).

For any subset L' of L, define *the auxiliary graph for* L', denoted by $\mathcal{G}(L')$, as the undirected graph with vertex set L' and edge set $E(L')$, where for every $(\{i, j\}, k) \in \mathcal{T}$ that satisfies $i, j, k \in L'$, the edge $\{i, j\}$ is included in $E(L')$. (Auxiliary graphs were first defined by Aho *et al.* [1].) To handle forbidden rooted triplets, for any subset L' of L, we introduce *the auxiliary partition* $\mathcal{D}(L')$ *of* L' as follows:

(1) Initially, let $\mathcal{D} = \{\mathcal{C}_1, \ldots, \mathcal{C}_q\}$ be a partition of L' such that each subset $\mathcal{C}_i$ consists of the set of nodes in one connected component of $\mathcal{G}(L')$.
(2) While there exists some $(\{i, j\}, k)$ in $\mathcal{F}|L'$ such that i and j are in one subset $S_1 \in \mathcal{D}$ and k is in another subset $S_2 \in \mathcal{D}$, merge S_1 and S_2 into one subset in $\mathcal{D}$.
(3) Set $\mathcal{D}(L') = \mathcal{D}$.

Our algorithm MTT proceeds recursively and is described in Figure 1.

Algorithm MTT
Input: A set $\mathcal{T}$ and a set $\mathcal{F}$ of rooted triplets and a leaf set L.
Output: A phylogenetic tree distinctly leaf-labeled by L that is consistent with all rooted triplets in $\mathcal{T}$ and no rooted triplets in $\mathcal{F}$, if one exists; otherwise *null*.
1 Construct the auxiliary partition $\mathcal{D}(L)$. Write $\mathcal{D}(L) = \{\mathcal{D}_1, \ldots, \mathcal{D}_r\}$.
2 If $r = 1$ and $\mathcal{D}_1$ consists of exactly one leaf i then return a tree with a single leaf labeled by i. If $r = 1$ and $\mathcal{D}_1$ contains more than one leaf then return *null*. Otherwise, for each $i \in \{1, \ldots, r\}$, let $T_i = MTT(\mathcal{T}|\mathcal{D}_i, \mathcal{F}|\mathcal{D}_i, \mathcal{D}_i)$; if all T_i-trees are not *null* then attach all of these trees to a common parent node and return the resulting tree, else return *null*.
End MTT

Figure 1. An algorithm for solving MTT.

Theorem 2.1. *Algorithm MTT outputs a phylogenetic tree T distinctly leaf-labeled by L that is consistent with all rooted triplets in $\mathcal{T}$ and no rooted triplets in $\mathcal{F}$, if and only if such a tree exists, in $O(|\mathcal{T}|\cdot n + |\mathcal{F}|\cdot n^2)$ time.*

Proof. If Algorithm MTT outputs a non-null tree T then by the correctness of the algorithm of Aho *et al.*, T is consistent with all rooted triplets in $\mathcal{T}$. Next, let $f = (\{i, j\}, k)$ be any rooted triplet in $\mathcal{F}$ and suppose that f is consistent with T. Then, at some recursion level of the algorithm where i, j, k are still in the same leaf set, i and j will belong to one subset $\mathcal{D}_a$ while k is in another subset $\mathcal{D}_b$. But this is impossible by step (2) in the construction of $\mathcal{D}(L)$. Contradiction; hence, f is not consistent with T.

Similarly, if the algorithm outputs *null* then at some recursion level, $\mathcal{D}(L)$ has just one element $\mathcal{D}_1$, where $\mathcal{D}_1$ contains at least two leaves. Suppose there exists a phylogenetic tree T^* that is consistent with all rooted triplets in $\mathcal{T}|\mathcal{D}_1$ and no rooted triplets in $\mathcal{F}|\mathcal{D}_1$. By the construction of $\mathcal{D}(L)$, two leaves in the same set $\mathcal{D}_a$ can not be descendants of two different children of the root of T^*. But since there is just one set $\mathcal{D}_1$, the root of T^* would only have one child, which is a contradiction. Hence, there is no phylogenetic tree that is consistent with all rooted triplets in $\mathcal{T}$ and no rooted triplets in $\mathcal{F}$.

There are $O(n)$ recursion levels, each of which is taken care of in $O(|\mathcal{T}|+|\mathcal{F}|\cdot n)$ time. Thus, the algorithm's total running time is $O((|\mathcal{T}| + |\mathcal{F}| \cdot n) \cdot n)$. □

2.2. *A polynomial-time approximation algorithm for MMTT*

MMTT restricted to $\mathcal{F} = \emptyset$ is NP-hard (see [5, 13, 28]), so it follows trivially that the unrestricted version of MMTT is NP-hard. Therefore, we provide a polynomial-time approximation algorithm for MMTT, which generalizes the following result from [9].

Lemma 2.1. [9] *Given a set $\mathcal{T}$ of rooted triplets with leaf set L, a phylogenetic tree distinctly leaf-labeled by L that is consistent with at least a third of the rooted triplets in $\mathcal{T}$ can be constructed in $O((|\mathcal{T}| + n) \cdot \log n)$ time.*

Theorem 2.2. *Given two sets $\mathcal{T}$ and $\mathcal{F}$ of rooted triplets with leaf set L, a phylogenetic tree T distinctly leaf-labeled by L such that $|T(\mathcal{T})| - |T(\mathcal{F})| \geq \frac{1}{3} \cdot (|\mathcal{T}| - |\mathcal{F}|)$ can be constructed in $O((|\mathcal{T}| + |\mathcal{F}| + n) \cdot \log n)$ time.*

Proof. We describe an algorithm to construct such a T. For every $v \in L$, keep a score associated with v, denoted by $s(v)$. Initially, set $s(v)$ to zero for every $v \in L$. Then, for every $(\{a,b\},c) \in \mathcal{T}$, increase $s(c)$ by one and decrease $s(a)$ and $s(b)$ by $\frac{1}{2}$, and for every $(\{a',b'\},c') \in \mathcal{F}$, decrease $s(c')$ by one and increase each of $s(a')$ and $s(b')$ by $\frac{1}{2}$. Next, assume $u \in L$ has the largest score. Let $L' = L \setminus \{u\}$. Then recursively construct a tree T' with leaf set L' such that $|T'(\mathcal{T}|L')| - |T'(\mathcal{F}|L')| \geq \frac{1}{3} \cdot (|\mathcal{T}|L'| - |\mathcal{F}|L'|)$ (if L' consists of only two leaves then let T' be a binary tree on those two leaves), attach the root of T' and a leaf labeled by u to a common parent node and let T be the resulting tree.

To prove the correctness of the algorithm, define $\mathcal{T}_u = \{t \mid t \in \mathcal{T} \text{ and } u \in \Lambda(t)\}$ and $\mathcal{F}_u = \{t \mid t \in \mathcal{F} \text{ and } u \in \Lambda(t)\}$. We now show that $|T(\mathcal{T}_u)| - |T(\mathcal{F}_u)| \geq \frac{1}{3} \cdot (|\mathcal{T}_u| - |\mathcal{F}_u|)$. Let $\mathcal{T}'_u = T(\mathcal{T}_u)$, $\mathcal{T}''_u = \mathcal{T}_u \setminus \mathcal{T}'_u$, $\mathcal{F}'_u = T(\mathcal{F}_u)$, and $\mathcal{F}''_u = \mathcal{F}_u \setminus \mathcal{F}'_u$. Then we can write $s(u) = |\mathcal{T}'_u| - \frac{1}{2} \cdot |\mathcal{T}''_u| - |\mathcal{F}'_u| + \frac{1}{2} \cdot |\mathcal{F}''_u|$. Since we choose a leaf u which has the largest score, it is easy to see that $s(u) \geq 0$. Therefore, $|\mathcal{T}'_u| - |\mathcal{F}'_u| \geq \frac{1}{2} \cdot (|\mathcal{T}''_u| - |\mathcal{F}''_u|)$. By adding $\frac{1}{2} \cdot (|\mathcal{T}'_u| - |\mathcal{F}'_u|)$ to both sides of the inequality and using $|\mathcal{T}'_u| + |\mathcal{T}''_u| = |\mathcal{T}_u|$ and $|\mathcal{F}'_u| + |\mathcal{F}''_u| = |\mathcal{F}_u|$, we obtain $|T(\mathcal{T}_u)| - |T(\mathcal{F}_u)| \geq \frac{1}{3} \cdot (|\mathcal{T}_u| - |\mathcal{F}_u|)$.

We can use a heap data structure to keep track of the changes in the scores of the leaves throughout the algorithm. The total running time becomes $O((|\mathcal{T}| + |\mathcal{F}| + n) \cdot \log n)$. □

Note that for the special case where $\mathcal{T}$ is empty, we have $|T(\mathcal{F})| \leq \frac{1}{3} \cdot |\mathcal{F}|$, i.e., the algorithm produces a tree that is consistent with at most one third of the rooted triplets in $\mathcal{F}$.

Also note that any phylogenetic tree produced by the algorithm above is always a binary tree (in fact, a binary caterpillar tree) and that any binary phylogenetic tree whose leaf set includes $\{a,b,c\}$ is consistent with exactly one of $(\{a,b\},c)$, $(\{a,c\},b)$, and $(\{b,c\},a)$. This means that if $\mathcal{T}$ and $\mathcal{F}$ have the property that $(\{a,b\},c) \in \mathcal{T}$ implies $(\{a,c\},b), (\{b,c\},a) \in \mathcal{T}$ and $(\{a',b'\},c') \in \mathcal{F}$ implies $(\{a',c'\},b'), (\{b',c'\},a') \in \mathcal{F}$ then $|T(\mathcal{T})| - |T(\mathcal{F})| = \frac{1}{3} \cdot (|\mathcal{T}| - |\mathcal{F}|)$ for any binary phylogenetic tree T with leaf set L. In this sense, the approximation algorithm is worst-case optimal.

3. NP-hardness of MT and a polynomial-time algorithm for a special case

3.1. *NP-hardness of MT*

To prove the NP-hardness of the general case of MT, we give a polynomial-time reduction from the following problem called *the forbidden rooted triplets problem* (FT): Given a set $\mathcal{S}$ of rooted triplets, is there a binary, rooted, distinctly leaf-labeled tree R that satisfies $\Lambda(R) = \Lambda(\mathcal{S})$ and $\mathcal{S} \cap \mathcal{R}(R) = \emptyset$? FT was shown to be NP-hard by Bryant [5].

Theorem 3.1. *MT is NP-hard.*

Proof. For any instance $\mathcal{S}$ of FT, construct an instance of MT by setting $\mathcal{T} = \emptyset$ and $\mathcal{F} = \mathcal{S}$. We claim that there exists a solution R for FT if and only if there exists a solution N for MT.

($\rightarrow$) If the first part of the statement holds, then the second part is trivially true because R is also a phylogenetic network.

($\leftarrow$) From N, we construct R as follows: For each recombination node in N, remove one of its two incoming edges, contract each outgoing edge from a node with resulting outdegree 1, and let R be the obtained tree. The claim follows because $\mathcal{R}(R) \subseteq \mathcal{R}(N)$. □

3.2. *An $O(n \log n)$-time algorithm for MT with disjoint forbidden rooted triplets*

Although MT is NP-hard, it can be solved efficiently if the forbidden rooted triplets are *disjoint* (meaning that $\Lambda(t) \cap \Lambda(t') = \emptyset$ for any $t, t' \in \mathcal{F}$ with $t \neq t'$), as we show in this section. Our algorithm is based on the following lemma from [16].

Lemma 3.1. [16] *For any set L of n leaf labels, a phylogenetic network N satisfying $\mathcal{R}(N) = \{(\{x, y\}, z) \mid x, y, z \in L\}$ can be constructed in $O(s(n))$ time, where $s(n)$ is the time required to construct a sorting network for n elements.*

By employing, e.g., an AKS sorting network (see [2]), we obtain $s(n) = O(n \log n)$ in Lemma 3.1 above. Now, suppose $\mathcal{F}$ is a given set of f disjoint forbidden rooted triplets and write $\mathcal{F} = \{(\{a_1, b_1\}, c_1), \ldots, (\{a_f, b_f\}, c_f)\}$. Let $P = \{p_1, q_1, \ldots, p_f, q_f\}$ be a set of labels not belonging to L. Build a phylogenetic network N as follows.

(1) Use Lemma 3.1 to construct a phylogenetic network N' which is consistent with all rooted triplets in $\{(\{x, y\}, z) \mid x, y, z \in (L \cup P \setminus \Lambda(\mathcal{F}))\}$ in $O(n \log n)$ time.
(2) For each $(\{a_i, b_i\}, c_i) \in \mathcal{F}$, make a_i be a child of p_i, b_i a child of q_i, and c_i a child of both p_i and q_i in N'. Let N be the resulting network.

Lemma 3.2. *For any $\mathcal{T} \subseteq \{(\{x, y\}, z) \mid x, y, z \in L\} \setminus \mathcal{F}$, it holds that $\mathcal{T} \subseteq \mathcal{R}(N)$. Furthermore, $\mathcal{F} \cap \mathcal{R}(N) = \emptyset$.*

Theorem 3.2. *MT with disjoint forbidden rooted triplets can be solved in $O(n \log n)$ time.*

4. Algorithms for MMTG and a restricted case of MTG

Here, we need the following additional terminology. Let N be a phylogenetic network and let h be a recombination node in N. Every ancestor s of h such that h can be reached using two disjoint directed paths starting at the children of s is called a *split node of h*. If s is a split node of h then any path starting at s and ending at h is called a *merge path of h* or a *merge path from s*. For any node u in N, $N[u]$ denotes the subnetwork of N rooted at u, i.e., the minimal subgraph of N which includes all nodes and directed edges of N reachable from u. $N[u]$ is called a *side network* of N if there exists a merge path P in N such that u does not belong to P but u is a child of a node belonging to P.

4.1. *A polynomial-time approximation algorithm for MMTG*

Jansson, Nguyen, and Sung [15] presented a $\frac{5}{12}$-approximation algorithm for MMTG restricted to $\mathcal{F} = \emptyset$. We can extend their algorithm to arbitrary $\mathcal{F}$, obtaining a polynomial-time algorithm for inferring a galled network N with $|N(\mathcal{T})| - |N(\mathcal{F})| \geq \frac{5}{12} \cdot (|\mathcal{T}| - |\mathcal{F}|)$.

The modified algorithm $MMTG(\mathcal{T}, \mathcal{F})$ is outlined in Figure 2. Similar to the original algorithm in [15], $MMTG$ first partitions the leaf set L into three subsets A, B, and C so that the value of a special score function $score(A, B, C) = 4(N_1 - M_1) + 7(N_2 - M_2) + 12(N_3 - M_3)$ is maximized, where for $i \in \{1, 2, 3\}$, we define two sets $X_i(A, B, C)$ and $Y_i(A, B, C)$ as below, and let $N_i = |X_i(A, B, C)|$ and $M_i = |Y_i(A, B, C)|$:

- $X_1(A, B, C) = \{(\{x, y\}, z) \in \mathcal{T} \mid x, y,$ and z are in one of the sets $A, B, C\}$,

- $X_2(A,B,C) = \{(\{x,y\},z) \in \mathcal{T} \mid x, y,$ and z are in three different sets$\}$, and
- $X_3(A,B,C) = \{(\{x,y\},z) \in \mathcal{T} \mid x$ and y are in one set and z is in another$\}$.

$Y_i(A,B,C)$ is defined analogously, using $\mathcal{F}$ instead of $\mathcal{T}$. We use a greedy algorithm to perform the partitioning that first randomly divides L into three arbitrary subsets and then keeps moving leaves from one subset to another until $score(A,B,C)$ cannot be further improved. After finished moving the leaves, if one of the sets, say A, equals L, we move the node u that maximizes $\frac{p_{\mathcal{T}}(u)-p_{\mathcal{F}}(u)}{c_{\mathcal{T}}(u)-c_{\mathcal{F}}(u)}$ from A to B, where $c_{\mathcal{G}}(u) = |\{(\{u,x\},y) \in \mathcal{G}\}|$ and $p_{\mathcal{G}}(u) = |\{(\{x,y\},u) \in \mathcal{G}\}|$ and $\mathcal{G} \in \{\mathcal{T},\mathcal{F}\}$. This extra step is to ensure that none of the three sets equals L. The rest of the algorithm proceeds as in [15] and recursively constructs three candidate galled networks N_A, N_B, and N_C. Finally, in Step 4, we return a network N_Z that maximizes $|N_Z(\mathcal{T})| - |N_Z(\mathcal{F})|$ among $Z \in \{A,B,C\}$.

Algorithm $MMTG$
Input: A set $\mathcal{T}$ and a set $\mathcal{F}$ of rooted triplets.
Output: A galled network N such that $|N(\mathcal{T})| - |N(\mathcal{F})| \geq \frac{5}{12} \cdot (|\mathcal{T}| - |\mathcal{F}|)$.
1 Partition $\Lambda(\mathcal{T}) \cup \Lambda(\mathcal{F})$ into three sets A, B, and C so that $score(A,B,C)$ is maximized.
2 For $Z \in \{A,B,C\}$, let $K_Z = MMTG(\mathcal{T}|Z, \mathcal{F}|Z)$.
3 Generate three galled networks N_A, N_B, N_C, where for instance, N_A is a galled network with side networks K_A, K_B, K_C such that K_A is attached to the recombination node h whose split node is the root, and K_B and K_C are attached to two different merge paths of h.
4 Among $Z \in \{A,B,C\}$, return a N_Z that maximizes $|N_Z(\mathcal{T})| - |N_Z(\mathcal{F})|$.
End $MMTG$

Figure 2. An approximation algorithm for MMTG.

Theorem 4.1. *A galled network N such that $|N(\mathcal{T})| - |N(\mathcal{F})| \geq \frac{5}{12} \cdot (|\mathcal{T}| - |\mathcal{F}|)$ can be constructed in $O(n \cdot (|\mathcal{T}| + |\mathcal{F}|)^3)$ time.*

For the correctness and time analysis of the algorithm, see [15].

4.2. *An $O(n^4)$-time algorithm for MTG when $\mathcal{T}$ is dense*

[15] also gave an $O(n^3)$-time algorithm for a restricted case of MTG where $\mathcal{T}$ is dense and $\mathcal{F} = \emptyset$. In this section we extend their algorithm to any $\mathcal{F}$.

Let u and v be two nodes in a galled network N. The subnetwork $N[v]$ is called a *direct subnetwork* of $N[u]$ if: (1) v is a child of u, if u is not a split node in N; or (2) v is attached to a merge path from u, if u is a split node.

A galled network N which is consistent with the rooted triplets in $\mathcal{T}$ and leaf-labeled by $L = \Lambda(\mathcal{T})$ is called *maximal* if there is no galled network M such that: (1) M is consistent with the triplets in $\mathcal{T}$ and leaf-labeled by L; and (2) there exists a direct subnetwork N' of N and a direct subnetwork M' of M such that $\Lambda(N') \subseteq \Lambda(M')$. N is *recursively maximal* if it is maximal and every direct subnetwork of it is recursively maximal. A galled network N which is consistent with the rooted triplets in $\mathcal{T}$ and leaf-labeled by $L = \Lambda(\mathcal{T})$ is called *nearly maximal* if either: (1) it is maximal; or (2) there exists a maximal galled network M which is consistent with the rooted triplets in $\mathcal{T}$, leaf-labeled by L, and whose root

is a split node with corresponding recombination node h such that N can be transformed from M in the following way: we remove one of the two edges pointing to h, say (g, h), and connect g with some new node obtained by subdividing some edge in $M[h]$ such that the new network is still a galled network. A subnetwork N is *recursively nearly maximal* if it is nearly maximal and every direct subnetwork of it is recursively nearly maximal.

For any dense set $\mathcal{T}$ of rooted triplets, if there exists a galled network consistent with $\mathcal{T}$ then there also exists a recursively maximal network consistent with $\mathcal{T}$ [15]. Given a dense set $\mathcal{T}$ of rooted triplets, the algorithm of [15] always output a recursively maximal network consistent with $\mathcal{T}$, if one exists. To deal with forbidden rooted triplets, we also need:

Lemma 4.1. *Given two sets of rooted triplets $\mathcal{T}$ and $\mathcal{F}$ where $\mathcal{T}$ is dense, if there exists a galled network that is consistent with $\mathcal{T}$ but not consistent with any rooted triplet in $\mathcal{F}$ then there also exists a recursively nearly maximal network that is so.*

Theorem 4.2. *Given two sets of rooted triplets $\mathcal{T}$ and $\mathcal{F}$ with leaf set $L = \Lambda(\mathcal{T}) \cup \Lambda(\mathcal{F})$ where $\mathcal{T}$ is dense, a galled network that is consistent with $\mathcal{T}$ but not consistent with any rooted triplet in $\mathcal{F}$ can be constructed in $O(n^4)$ time, where $n = |L|$.*

Proof. (Sketch.) The algorithm looks for a recursively nearly maximal network that is consistent with all rooted triplets in $\mathcal{T}$ but none in $\mathcal{F}$. The algorithm in [15] finds a recursively maximal galled network consistent with $\mathcal{T}$ in $O(n^3)$ time, if one exists. We can modify it to return a recursively nearly maximal galled network consistent with $\mathcal{T}$ but not consistent with any rooted triplets in $\mathcal{F}$ by exploiting the fact that any nearly maximal network can be transformed from some maximal network. Overall, the time taken is $O(n^4)$. □

References

1. A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.
2. M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. In *Proceedings of the 15thAnnual ACM Symposium on the Theory of Computing* (STOC'83), pages 1–9, 1983.
3. V. Berry and F. Nicolas. Maximum agreement and maximum compatible supertrees. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching* (CPM 2004), pages 205–219, 2004.
4. O. Bininda-Emonds, J. Gittleman, and M. Steel. The (super)tree of life: Procedures, problems, and prospects. *Annual Review of Ecology and Systematics*, 33:265–289, 2002.
5. D. Bryant. *Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis*. PhD thesis, University of Canterbury, Christchurch, New Zealand, 1997.
6. B. Chor, M. Hendy, and D. Penny. Analytic solutions for three-taxon ML$_{MC}$ trees with variable rates across sites. In *Proceedings of the 1st Workshop on Algorithms in Bioinformatics* (WABI 2001), volume 2149 of *LNCS*, pages 204–213. Springer-Verlag, 2001.
7. C. Choy, J. Jansson, K. Sadakane, and W.-K. Sung. Computing the maximum agreement of phylogenetic networks. In *Proceedings of Computing: the 10th Australasian Theory Symposium* (CATS 2004), pages 33–45. Elsevier, 2004.
8. L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. Inferring ordered trees from local constraints. In *Proceedings of Computing: the 4th Australasian Theory Symposium* (CATS'98), volume

20(3) of *Australian Computer Science Communications*, pages 67–76. Springer-Verlag Singapore, 1998.
9. L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. On the complexity of constructing evolutionary trees. *Journal of Combinatorial Optimization*, 3(2–3):183–197, 1999.
10. D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proceedings of the Computational Systems Bioinformatics Conference* (CSB2003), pages 363–374, 2003.
11. M. R. Henzinger, V. King, and T. Warnow. Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. *Algorithmica*, 24(1):1–13, 1999.
12. D. H. Huson, T. Dezulian, T. Klöpper, and M. Steel. Phylogenetic super-networks from partial trees. In *Proceedings of the 4^{th} Workshop on Algorithms in Bioinformatics* (WABI 2004), to appear.
13. J. Jansson. On the complexity of inferring rooted evolutionary trees. In *Proceedings of the Brazilian Symposium on Graphs, Algorithms, and Combinatorics* (GRACO 2001), volume 7 of *Electronic Notes in Discrete Mathematics*, pages 121–125. Elsevier, 2001.
14. J. Jansson, J. H.-K. Ng, K. Sadakane, and W.-K. Sung. Rooted maximum agreement supertrees. In *Proceedings of Latin American Theoretical Informatics* (LATIN 2004), volume 2976 of *LNCS*, pages 499–508. Springer-Verlag, 2004.
15. J. Jansson, N. B. Nguyen, and W.-K. Sung. Algorithms for combining rooted triplets into a galled phylogenetic network. In *Proceedings of the 16^{th} Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA 2005), to appear.
16. J. Jansson and W.-K. Sung. Inferring a level-1 phylogenetic network from a dense set of rooted triplets. In *Proceedings of the 10^{th} International Computing and Combinatorics Conference* (COCOON 2004), volume 3106 of *LNCS*, pages 462–471. Springer-Verlag, 2004.
17. S. Kannan, E. Lawler, and T. Warnow. Determining the evolutionary tree using experiments. *Journal of Algorithms*, 21(1):26–50, 1996.
18. P. Kearney. Phylogenetics and the quartet method. In T. Jiang, Y. Xu, and M. Q. Zhang, editors, *Current Topics in Computational Molecular Biology*, pages 111–133. The MIT Press, Massachusetts, 2002.
19. W.-H. Li. *Molecular Evolution*. Sinauer Associates, Inc., Sunderland, 1997.
20. C. R. Linder, B. M. E. Moret, L. Nakhleh, and T. Warnow. Network (reticulate) evolution: Biology, models, and algorithms. Tutorial presented at *the 9^{th} Pacific Symposium on Biocomputing* (PSB 2004), 2004.
21. L. Nakhleh, T. Warnow, and C. R. Linder. Reconstructing reticulate evolution in species – theory and practice. In *Proceedings of the 8^{th} Annual International Conference on Research in Computational Molecular Biology* (RECOMB 2004), pages 337–346, 2004.
22. M. P. Ng, M. Steel, and N. C. Wormald. The difficulty of constructing a leaf-labelled tree including or avoiding given subtrees. *Discrete Applied Mathematics*, 98(3):227–235, 2000.
23. D. Posada and K. A. Crandall. Intraspecific gene genealogies: trees grafting into networks. *TRENDS in Ecology & Evolution*, 16(1):37–45, 2001.
24. M. J. Sanderson, A. Purvis, and C. Henze. Phylogenetic supertrees: assembling the trees of life. *TRENDS in Ecology & Evolution*, 13(3):105–109, 1998.
25. J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, 1997.
26. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.
27. L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.
28. B. Y. Wu. Constructing the maximum consensus tree from rooted triples. *Journal of Combinatorial Optimization*, 8:29–39, 2004.