

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: {heyjingju, huynhnngo, 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{C} 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 nodes 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, 14, 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

*Wing-Kin 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 a phylogenetic network from a given set \mathcal{C} 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 to as *hybrid 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 and have attracted special attention in the literature [7, 10, 21, 27] due to their biological significance and their simple, almost treelike, structure. When the number of recombination events is limited and most of the recombination events have occurred recently, a galled network may suffice to accurately describe the evolutionary process under study [10]. Galled networks 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{C} of rooted triplets is *consistent with N* if every rooted triplet in \mathcal{C} is consistent with N . The set of all rooted triplets which are consistent with N is denoted by $\mathcal{R}(N)$, and we let $N(\mathcal{C})$ be the subset of \mathcal{C} containing all rooted triplets from \mathcal{C} that are consistent with N , i.e., $N(\mathcal{C}) = \mathcal{C} \cap \mathcal{R}(N)$.

Denote the set of leaves in a phylogenetic tree/network N by $\Lambda(N)$, and for any set \mathcal{C} of rooted triplets, define $\Lambda(\mathcal{C}) = \bigcup_{t \in \mathcal{C}} \Lambda(t)$. Given a leaf set L , a set \mathcal{C} of rooted triplets is called *dense (with respect to L)* if $\Lambda(\mathcal{C}) = L$ 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{C} . Finally,

for any set \mathcal{C} of rooted triplets and $L' \subseteq \Lambda(\mathcal{C})$, we define $\mathcal{C} \upharpoonright L'$ as the subset of \mathcal{C} consisting of all rooted triplets $(\{x, y\}, z)$ with $\{x, y, z\} \subseteq L'$.

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

- *The mixed triplets problem (MT)*: Construct a phylogenetic network N with $\Lambda(N) = L$ such that $\mathcal{C} \subseteq \mathcal{R}(N)$ 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 $\mathcal{C} \subseteq \mathcal{R}(T)$ 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 $O(|\mathcal{C}| \cdot n + |\mathcal{F}| \cdot n \log n + n^2 \log n)$ -time algorithm for MTT. We also study a corresponding maximization problem that we call MMTT which asks for a phylogenetic tree T that maximizes $|T(\mathcal{C})| - |T(\mathcal{F})|$.^a MMTT is NP-hard^b, so we present a polynomial-time algorithm for inferring a phylogenetic tree T that guarantees $|T(\mathcal{C})| - |T(\mathcal{F})| \geq \frac{1}{3} \cdot (|\mathcal{C}| - |\mathcal{F}|)$ in Section 2.2.
- *The mixed triplets problem restricted to galled networks (MTG)*: Construct a galled network N with $\Lambda(N) = L$ such that $\mathcal{C} \subseteq \mathcal{R}(N)$ 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, we study the maximization version of MTG called MMTG and give a polynomial-time algorithm for inferring a galled network N that guarantees $|N(\mathcal{C})| - |N(\mathcal{F})| \geq \frac{5}{12} \cdot (|\mathcal{C}| - |\mathcal{F}|)$.

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 $O(|\mathcal{C}| \cdot n)$ -time algorithm for determining whether a given set \mathcal{C} 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 $\mathcal{F} = \emptyset$)^c. 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

^aThis problem is useful when there is no phylogenetic tree T such that $\mathcal{C} \subseteq \mathcal{R}(T)$ and $\mathcal{F} \cap \mathcal{R}(T) = \emptyset$.

^bWhen $\mathcal{F} = \emptyset$, Bryant [5] showed that this problem is NP-hard, so the problem is also NP-hard for any \mathcal{F} .

^cIn 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].

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 $\Lambda(\mathcal{C})$ such that $\mathcal{C} \upharpoonright L'$ 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 phylogenetic tree which is consistent with at least $\frac{1}{3} \cdot |\mathcal{C}|$ of the rooted triplets in \mathcal{C}).

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 hybrid 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 case $\mathcal{F} \neq \emptyset$, much less is known. Bryant [5] showed that MTT restricted to $\mathcal{C} = \emptyset$ 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].

1.3. The algorithm of Aho, Sagiv, Szymanski, and Ullman

Here, we briefly review the algorithm of Aho *et al.* [1] for determining if a given set \mathcal{C} of rooted triplets with leaf set L is consistent with a rooted tree, and if so, constructing one. (Please refer to [1] for correctness proofs.) For any subset L' of the leaves in 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 $(\{i, j\}, k) \in \mathcal{C}$ that satisfies $i, j, k \in L'$, the edge $\{i, j\}$ is included in $E(L')$.

Given a set \mathcal{C} , the algorithm of Aho *et al.* builds $\mathcal{G}(L)$ and calculates the connected components A_1, \dots, A_q of $\mathcal{G}(L)$. If $q \geq 2$, the algorithm recursively constructs a tree for each connected component, attaches these trees to a common parent node, and returns the resulting tree; the q recursive calls to itself are done on the sets $\mathcal{C}_1, \dots, \mathcal{C}_q$ obtained by

scanning \mathcal{C} (for $1 \leq p \leq q$, let L_p be the set of leaves in A_p , and for each $(\{i, j\}, k) \in \mathcal{C}$, if $\{i, j, k\} \subseteq L_p$ then $(\{i, j\}, k)$ is placed in \mathcal{C}_p ; otherwise it is deleted). Otherwise, there is just one connected component A_1 . If A_1 consists of a single leaf i , the algorithm returns a tree with one leaf labeled by i . If A_1 contains more than one leaf, the algorithm aborts its execution and returns *null* since no tree can be consistent with all of the rooted triplets in this case.

At each of the $O(n)$ recursion levels, the total time required to build all auxiliary graphs and to find their connected components is $O(|\mathcal{C}|)$. Scanning the rooted triplets to compute the sets \mathcal{C}_p also takes $O(|\mathcal{C}|)$ time on each level. Therefore, this implementation has a running time of $O(|\mathcal{C}| \cdot n)$.

2. Algorithms for MTT and MMTT

In Section 2.1, we present a polynomial time algorithm for MTT. Then in Section 2.2, we describe an approximation algorithm for the corresponding maximization problem MMTT.

2.1. A polynomial time algorithm for MTT

Our algorithm is a generalization of the algorithm of Aho *et al.* [1] (see Section 1.3) for determining if a given set \mathcal{C} 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 $\mathcal{C} = \emptyset$ (see Section 1.2).

To handle forbidden rooted triplets, for any subset L' of L , we define the *auxiliary partition* $\mathcal{D}(L')$ of L' as follows:

- i. Initially, let $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_q\}$ be a partition of L' such that each subset \mathcal{D}_i consists of the set of nodes in one connected component of the auxiliary graph $\mathcal{G}(L')$ for L' (the auxiliary graph is defined in Section 1.3).
- ii. While there exists some $(\{i, j\}, k)$ in $\mathcal{F}|L'$ such that i and j are in one subset $\mathcal{D}_a \in \mathcal{D}$ and k is in another subset $\mathcal{D}_b \in \mathcal{D}$, let $\mathcal{D} = (\mathcal{D} \setminus \{\mathcal{D}_a, \mathcal{D}_b\}) \cup \{\mathcal{D}_a \cup \mathcal{D}_b\}$.
- iii. Set $\mathcal{D}(L') = \mathcal{D}$.

Our algorithm proceeds as follows. Given \mathcal{C} and \mathcal{F} , we build $\mathcal{D}(L)$ for L . Let $\mathcal{D}(L) = \{\mathcal{D}_1, \dots, \mathcal{D}_r\}$. If $r \geq 2$, we recursively construct a tree for each subset \mathcal{D}_i , then attach these trees to a common parent node, and return the resulting tree. Otherwise, there is just one component \mathcal{D}_1 . If \mathcal{D}_1 consists of a single leaf i , we return a tree which is a leaf labeled by i . If \mathcal{D}_1 contains more than one leaf, we return *null* and conclude that there is no tree that is consistent with all of the rooted triplets in \mathcal{C} and none of the rooted triplet in \mathcal{F} . The algorithm is shown in Figure 1.

Lemma 1. *Let T be any tree that is consistent with all rooted triplets in \mathcal{C} and no rooted triplet in \mathcal{F} . Then two leaves in the same set in $\mathcal{D}(L)$ cannot be descendants of two different children of the root of T .*

Algorithm *MTT***Input:** A set \mathcal{C} 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 of the rooted triplets in \mathcal{C} and none of the rooted triplets in \mathcal{F} , if one exists; otherwise *null*.**1** Construct the auxiliary partition $\mathcal{D}(L)$. Write $\mathcal{D}(L) = \{\mathcal{D}_1, \dots, \mathcal{D}_r\}$.**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 $i \in \{1, \dots, r\}$, let $T_i = \text{MTT}(\mathcal{C}|_{\mathcal{D}_i}, \mathcal{F}|_{\mathcal{D}_i}, \mathcal{D}_i)$; 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*

Fig. 1. An exact algorithm for solving MTT.

Proof. We prove by induction on the construction of $\mathcal{D}(L)$. Right after Step (i) in the construction, two leaves in the same set in \mathcal{D} belong to the same connected component in $\mathcal{G}(L)$. Therefore, by the correctness of Aho *et al.* algorithm, they cannot be descendants of two different children of the root of T ; otherwise, T would be not consistent with all rooted triplets in \mathcal{C} .

At Step (ii), suppose that we are considering a rooted triplet $(\{i, j\}, k)$ in \mathcal{F} such that i and j belong to the same set \mathcal{D}_a in \mathcal{D} and k belongs to a different set \mathcal{D}_b and hence, we are going to merge \mathcal{D}_a and \mathcal{D}_b into one single set. By induction, leaves i and j (and all other leaves in \mathcal{D}_a) must be descendants of a child c of the root of \mathcal{C} . If k is not a descendant of c in T , then T would be consistent with $(\{i, j\}, k)$. Hence, k is also a descendant of c and so are all other leaves in \mathcal{D}_b . Thus, all leaves in \mathcal{D}_a and \mathcal{D}_b are descendants of the same children of the root of T . \square

Lemma 2. Suppose there exists a tree T that is consistent with all rooted triplets in \mathcal{C} and no rooted triplet in \mathcal{F} . Let L' be any nonempty subset of L , then there exists a tree T' that is consistent with all rooted triplets in $\mathcal{C}|_{L'}$ and no rooted triplet in $\mathcal{F}|_{L'}$.

Proof. Let $T' = T|_{L'}$. It is easy to see that a rooted triplet t with $\Lambda(t) \subseteq L'$ is consistent with T if and only if it is consistent with T' . Then all rooted triplets in $\mathcal{C}|_{L'}$ are consistent with T' while no rooted triplet in $\mathcal{F}|_{L'}$ is consistent with T' . \square

Lemma 3. The auxiliary partition $\mathcal{D}(L)$ can be constructed in $O(|\mathcal{C}| + |\mathcal{F}|\log n + n \log n)$ time.

Proof. Step (i) of the construction of $\mathcal{D}(L)$ can be done in $O(|\mathcal{C}|)$ time as in the algorithm of Aho *et al.* [1] (see Section 1.3). To do Step (ii), we maintain a set S containing a subset of \mathcal{F} such that $S = \{(\{x, y\}, z) \mid (\{x, y\}, z) \in \mathcal{F} \text{ and } x \text{ and } y \text{ are in one subset } \mathcal{D}_a \in \mathcal{D} \text{ and } z \text{ is in another subset } \mathcal{D}_b \in \mathcal{D}\}$. While S is nonempty, we choose any triplet $(\{x, y\}, z)$ in S , combine the two corresponding subsets in \mathcal{D} that contain x, y , and z , and update S accordingly. When S is empty, Step (ii) is done.

To maintain S in Step (ii), we just need to update S whenever we combine two subsets in \mathcal{D} into a single set, since this is the only time that rooted triplets in \mathcal{F} may need to be removed from or inserted into S . Associated with each leaf u in L is a list $L(u)$ of all forbidden rooted triplets in \mathcal{F} that u belongs to (thus, every forbidden triplet occurs in three different lists and the total length of all lists is $O(|\mathcal{F}|)$). When two subsets \mathcal{D}_a and \mathcal{D}_b are to be combined, we update all leaves in the *smaller* of the two subsets, say \mathcal{D}_a , so that they all belong to \mathcal{D}_b . Moreover, by traversing $L(u)$ for all u that belonged to \mathcal{D}_a , we locate every rooted triplet in \mathcal{F} that might need to be removed from or inserted into S , and update S accordingly. Every leaf is updated at most $O(\log n)$ times in total since we always update the smaller subset, and hence, every list is traversed at most $O(\log n)$ times throughout the construction. Therefore, Step (ii) can be done in $O(|\mathcal{F}| \cdot \log n + n \log n)$ time.

The lemma follows. \square

Theorem 1. *Algorithm MTT outputs a phylogenetic tree T distinctly leaf-labeled by L that is consistent with all rooted triplets in \mathcal{C} and no rooted triplets in \mathcal{F} , if one exists, in $O(|\mathcal{C}| \cdot n + |\mathcal{F}| \cdot n \log n + n^2 \log n)$ 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{C} (since for any $L' \subseteq L$, the set of nodes in any connected component of $\mathcal{G}(L')$ is a subset of some set in $\mathcal{D}(L')$). Next, let $f = (\{i, j\}, k)$ be any rooted triplet in \mathcal{F} and assume 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 (ii) in the construction of $\mathcal{D}(L)$. Hence, we arrive at contradiction and 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^* such that all rooted triplets in $\mathcal{C}|\mathcal{D}_1$ are consistent with T^* while no rooted triplets in $\mathcal{F}|\mathcal{D}_1$ is consistent with T^* . By Lemma 1, two leaves in the same set \mathcal{D}_a cannot 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 such T^* . By Lemma 2, there is no phylogenetic tree that is consistent with all rooted triplets in \mathcal{C} and no rooted triplets in \mathcal{F} .

There are $O(n)$ recursion levels, each of which is taken care of in $O(|\mathcal{C}| + |\mathcal{F}| \cdot \log n + n \log n)$ time by Lemma 3. Thus, the algorithm's total running time is $O(|\mathcal{C}| \cdot n + |\mathcal{F}| \cdot n \log n + n^2 \log n)$. \square

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 4. [9] *Given a set \mathcal{C} of rooted triplets with leaf set L , a phylogenetic tree distinctly*

leaf-labeled by L that is consistent with at least one third of the rooted triplets in \mathcal{C} can be constructed in $O((|\mathcal{C}| + n) \cdot \log n)$ time.

Theorem 2. *Given two sets \mathcal{C} and \mathcal{F} of rooted triplets with leaf set L , a phylogenetic tree T distinctly leaf-labeled by L such that $|T(\mathcal{C})| - |T(\mathcal{F})| \geq \frac{1}{3} \cdot (|\mathcal{C}| - |\mathcal{F}|)$ can be constructed in $O((|\mathcal{C}| + |\mathcal{F}| + n) \cdot \log n)$ time.*

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

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

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 $O((|\mathcal{C}| + |\mathcal{F}| + n) \cdot \log n)$. \square

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 $\mathcal{C} = \{(\{a, b\}, c), (\{a, c\}, b), (\{b, c\}, a) \mid (\{a, b\}, c) \in S\}$ and $\mathcal{F} = \{(\{a, b\}, c), (\{a, c\}, b), (\{b, c\}, a) \mid (\{a, b\}, c) \in S'\}$, where $S, S' \subseteq \{(\{a, b\}, c) \mid a, b, c \in L\}$, then $|T(\mathcal{C})| = \frac{1}{3}|\mathcal{C}|$ and $|T(\mathcal{F})| = \frac{1}{3}|\mathcal{F}|$. Hence, $|T(\mathcal{C})| - |T(\mathcal{F})| = \frac{1}{3} \cdot (|\mathcal{C}| - |\mathcal{F}|)$ for any binary phylogenetic tree T with leaf set L . In this sense, the approximation algorithm is worst-case optimal.^d

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

^dBryant [5] shows that the mixed triplet problem is NP-hard when $\mathcal{C} = \emptyset$ and the phylogenetic structure is restricted to binary tree. Hence this restricted case is also NP-hard for any \mathcal{C} .

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

Section 3.1 shows the NP-completeness of MT. Then in Section 3.2, we show that a restricted case of MT can be solved efficiently.

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 T that satisfies $\Lambda(T) = \Lambda(\mathcal{S})$ and $\mathcal{S} \cap \mathcal{R}(T) = \emptyset$? FT was shown to be NP-hard by Bryant [5].

Theorem 3. *MT is NP-hard.*

Proof. For any instance \mathcal{S} of FT, construct an instance of MT by setting $\mathcal{C} = \emptyset$ and $\mathcal{F} = \mathcal{S}$. We claim that there exists a solution T 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 T is also a phylogenetic network.

(\leftarrow) From N , we construct T as follows: For each hybrid node in N , remove one of its two incoming edges. Then, for each node with outdegree 1 and indegree less than 2, contract its outgoing edge. Let T be the obtained tree. It is easy to see that $\mathcal{R}(T) \subseteq \mathcal{R}(N)$. Then since N is consistent with no triplet in \mathcal{F} , so is T and the claim follows. \square

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

Although MT is NP-hard, this section shows that 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 $t \neq t'$). Our algorithm is based on the following lemma from [16].

Lemma 5. [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 5 above. Now, suppose \mathcal{F} is a given set of f disjoint forbidden rooted triplets and write $\mathcal{F} = \{(\{a_1, b_1\}, c_1), \dots, (\{a_f, b_f\}, c_f)\}$. Let $P = \{p_1, q_1, \dots, p_f, q_f\}$ be a set of labels not belonging to L . We build a phylogenetic network N as follows.

- (1) Use Lemma 5 to construct a phylogenetic network N' which is consistent with all rooted triplets in $\{(\{x, y\}, z) \mid x, y, z \in ((L \setminus \Lambda(\mathcal{F})) \cup P)\}$ 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.

Then, N can be constructed in $O(n \log n)$ time and has the following property.

Lemma 6. *For any $\mathcal{C} \subseteq \{(\{x, y\}, z) \mid x, y, z \in L\} \setminus \mathcal{F}$, $\mathcal{C} \subseteq \mathcal{R}(N)$. Furthermore, $\mathcal{F} \cap \mathcal{R}(N) = \emptyset$.*

Proof. Let t be any rooted triplet in the set \mathcal{C} . There are three possible cases.

- $|\Lambda(t) \cap \{a_i, b_i, c_i\}| \leq 2$ for any $1 \leq i \leq |\mathcal{F}|$. Note that, by the construction of N , for any $S \subset \{a_i, b_i, c_i\}$ and $|S| \leq 2$, there are always $|S|$ disjoint paths from $\{p_i, q_i\}$ to S in N . Since any rooted triplet on the leaf set $(L \setminus \Lambda(\mathcal{F})) \cup P$ is consistent with N' , t is consistent with N .
- $t = (\{b_i, c_i\}, a_i)$ for some $1 \leq i \leq |\mathcal{F}|$. Since both b_i and c_i are children of q_i and a_i is a child of p_i , t is consistent with N .
- $t = (\{a_i, c_i\}, b_i)$ for some $1 \leq i \leq |\mathcal{F}|$. Since both a_i and c_i are children of p_i and b_i is a child of q_i , t is consistent with N .

Since $t \in \mathcal{R}(N)$ for any $t \in \mathcal{C}$, we have $\mathcal{C} \subseteq \mathcal{R}(N)$.

Next, we show that $\mathcal{F} \cap \mathcal{R}(N) = \emptyset$. Consider any $(\{a_i, b_i\}, c_i) \in \mathcal{F}$. By the construction of N , the lowest common ancestor of a_i and b_i is the lowest common ancestor of p_i and q_i . Since p_i and q_i are the only parents of c_i , the lowest common ancestor of a_i and b_i cannot be a descendant of that of a_i and c_i . Hence, $(\{a_i, b_i\}, c_i)$ is not consistent with N . \square

Hence, we have the following result.

Theorem 4. *MT with disjoint forbidden triplets can be solved in $O(n \log n)$ time.*

4. A polynomial-time approximation algorithm for MMTG

Here, we need the following additional terminology. Let N be a phylogenetic network and let h be a hybrid 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* . Observe that in a galled network, each split node is a split node of exactly one hybrid node, and each hybrid node has exactly one split node. 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 . In this case, $N[u]$ is also said to be *attached to P* .

Jansson, Nguyen and Sung [15] proposed a $\frac{5}{12}$ -approximation algorithm for a restricted case of MMTG where $\mathcal{F} = \emptyset$. In this section we extend their algorithm to arbitrary \mathcal{F} . Given two sets of rooted triplets \mathcal{C} and \mathcal{F} , this section presents an algorithm for inferring a galled phylogenetic network N such that $|N(\mathcal{C})| - |N(\mathcal{F})| \geq \frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|)$. We describe the algorithm first and then present the analysis.

Similar to the original algorithm in [15], our algorithm $MMTG(\mathcal{C}, \mathcal{F})$ is recursive in nature. It is shown in Fig. 2. First, it partitions the set of leaves L into 3 subsets A, B, C such that none of them equals L using algorithm *LeafPartition* (also shown in Fig. 2 and described in detail later). Then, for each $X \in \{A, B, C\}$, it recursively infers a galled network K_X by calling $MMTG(\mathcal{C}|X, \mathcal{F}|X)$. Next, for each $X \in \{A, B, C\}$, it generates a galled network $Network_X$ such that the root node is a split node whose hybrid node is the parent of K_X , and the other two networks in $\{K_A, K_B, K_C\} \setminus \{K_X\}$ are side networks

attached to both side of the merge paths of h . Finally, we return the network $Network_Z$ that maximizes $|Network_Z(\mathcal{C})| - |Network_Z(\mathcal{F})|$.

To partition L into three sets A , B , and C in Step 1 of the algorithm, we partition L so that a special condition $4(N_1 - M_1) + 7(N_2 - M_2) + 12(N_3 - M_3) \geq 5 \cdot (|\mathcal{C}| - |\mathcal{F}|)$ holds, where for $i \in \{0, 1, 2, 3\}$, we define $N_i = |X_i(A, B, C)|$ and $M_i = |Y_i(A, B, C)|$ and where $X_i(A, B, C)$ and $Y_i(A, B, C)$ are sets defined as follows:

- $X_0(A, B, C) = \{(\{x, y\}, z) \in \mathcal{C} | x \text{ and } z \text{ are in one set and } y \text{ is in another or } y \text{ and } z \text{ are in one set and } x \text{ is in another}\},$
- $X_1(A, B, C) = \{(\{x, y\}, z) \in \mathcal{C} | x, y \text{ and } z \text{ are in one set}\},$
- $X_2(A, B, C) = \{(\{x, y\}, z) \in \mathcal{C} | x, y \text{ and } z \text{ are in three different sets}\}, \text{ and}$
- $X_3(A, B, C) = \{(\{x, y\}, z) \in \mathcal{C} | x \text{ and } y \text{ are in one set and } z \text{ is in another}\}.$

$Y_i(A, B, C)$ is defined analogously, using \mathcal{F} instead of \mathcal{C} . We use a greedy algorithm to perform the partitioning. It first randomly divides L into three arbitrary subsets. Then, the algorithm keeps moving leaves from one subset to another until the score $score(A, B, C)$ cannot be further improved, where we define $score(A, B, C) = 4(N_1 - M_1) + 7(N_2 - M_2) + 12(N_3 - M_3)$. After finished moving the leaves, if one of the set, says A , equals L , we move the node u that maximizes $\frac{p_{\mathcal{C}}(u) - p_{\mathcal{F}}(u)}{c_{\mathcal{C}}(u) - c_{\mathcal{F}}(u)}$ 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 $\mathcal{G} \in \{\mathcal{C}, \mathcal{F}\}$. This step is to ensure that none of the three sets equals L . The algorithm is called Algorithm *LeafPartition* and is also shown in Fig. 2.

The rest of this section analyzes the Algorithm *MMTG*. The lemma below analyzes the greedy partitioning algorithm.

Lemma 7. *Algorithm LeafPartition partitions the set L into three subsets A, B, C such that $score(A, B, C)$ cannot be further improved by moving exactly one element from one subset to another.*

Proof. After Step 2 of Algorithm *LeafPartition*, if none of the 3 subsets equals L , the lemma follows. Assume that after Step 2, one of the subsets, says A , equals L . We only need to show that Step 3 does not decrease $score(A, B, C)$. When u is moved from A to B , all triplets in $\{(\{u, x\}, y) \in \mathcal{C}\}$ are moved from X_1 to X_0 and all triplets in $\{(\{x, y\}, u) \in \mathcal{C}\}$ are moved from X_1 to X_3 . Similarly, all triplets in $\{(\{u, x\}, y) \in \mathcal{F}\}$ are moved from Y_1 to Y_0 and all triplets in $\{(\{x, y\}, u) \in \mathcal{F}\}$ are moved from Y_1 to Y_3 . Thus $score(A - \{u\}, \{u\}, \emptyset) - score(A, \emptyset, \emptyset) = (p_{\mathcal{C}}(u) - p_{\mathcal{F}}(u))(12 - 4) - (c_{\mathcal{C}}(u) - c_{\mathcal{F}}(u))4$. Given the fact that $p_{\mathcal{C}}(u) - p_{\mathcal{F}}(u) \geq 1/2(c_{\mathcal{C}}(u) - c_{\mathcal{F}}(u))$, Step 3 does not decrease $score(A, B, C)$. \square

Lemma 8. *When Algorithm MMTG finishes the partitioning in Step 1, we have $\frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|) \leq \frac{5}{12}(N_1 - M_1) + \frac{2}{3}(N_2 - M_2) + (N_3 - M_3)$.*

Proof. Let $score(A, B, C) = x(N_1 - M_1) + y(N_2 - M_2) + z(N_3 - M_3)$. When algorithm *LeafPartition* terminates, for any $\{U, V\} \subseteq \{A, B, C\}$, moving a leaf m from U to V cannot increase $score(A, B, C)$. Considering moving m from A to B , we can deduce a formula to compute the change of score with respect to \mathcal{C} and \mathcal{F} . Since moving m cannot

Algorithm *LeafPartition***Input:** Sets of rooted triplets \mathcal{C} and \mathcal{F} with leaf set L .**Output:** A partition of L into 3 sets A, B, C such that none of them equals L and such that $4(N_1 - M_1) + 7(N_2 - M_2) + 12(N_3 - M_3) \geq 5 \cdot (|\mathcal{C}| - |\mathcal{F}|)$.

```

1 Arbitrarily partition  $L$  into 3 sets  $A, B, C$ .
2 while moving an element  $m$  from one set to another increases  $score(A, B, C)$  do
2.1 move  $m$  accordingly.
    endwhile
3 if one of the subsets, says,  $A$  equals  $L$  then
3.1 choose  $u \in A$  that maximizes  $\frac{pc(u) - pf(u)}{cc(u) - cf(u)}$  and move it to  $B$ .
3.2 go to Step 2.
    else
3.3 return  $A, B, C$ .
    endif
End LeafPartition

```

Algorithm *MMTG***Input:** Sets of rooted triplets \mathcal{C} and \mathcal{F} **Output:** A galled network N such that $|N(\mathcal{C})| - |N(\mathcal{F})| \geq \frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|)$

```

1 Partition  $\Lambda(\mathcal{C}) \cup \Lambda(\mathcal{F})$  into  $A, B, C$  by LeafPartition
2 Let  $K_Z = MMTG(\mathcal{C}|Z, \mathcal{F}|Z)$  for  $Z \in \{A, B, C\}$ 
3 For  $Z \in \{A, B, C\}$ , generate a galled network  $Network_Z$  in which the root node is
  a split node whose hybrid node  $h$  is the parent of  $K_Z$ , and the other two networks in
   $\{K_A, K_B, K_C\} \setminus \{K_Z\}$  are side networks attached to both sides of the merge paths of  $h$ .
4 Among  $Z \in \{A, B, C\}$ , return  $Network_Z$  with the largest  $|Network_Z(\mathcal{C})| - |Network_Z(\mathcal{F})|$ .
End MMTG

```

Fig. 2. An approximation algorithm for MMTG.

increase the score, the change of score must be nonpositive. By summing all possible ways of moving m , we can derive the inequality $(2z - 6x)(N_1 - M_1) + (z + 2y + x)(N_0 - M_0) + (2z - 6y)(N_2 - M_2) + (2y + x - 5z)(N_3 - M_3) \leq 0$. By substituting $N_0 = |\mathcal{C}| - N_1 - N_2 - N_3$ and $M_0 = |\mathcal{F}| - M_1 - M_2 - M_3$ and replacing $x = 4, y = 7, z = 12$, we have $5(|\mathcal{C}| - |\mathcal{F}|) \leq 5(N_1 - M_1) + 8(N_2 - M_2) + 12(N_3 - M_3)$. \square

Let denote $N_{L'}$ be the galled network returned by $MMTG(\mathcal{C}|L', \mathcal{F}|L')$.

Lemma 9. *If $|N_X(\mathcal{C}|X)| - |N_X(\mathcal{F}|X)| \geq t(|\mathcal{C}|X| - |\mathcal{F}|X|) \forall X \in \{A, B, C\}$ then $|N_L(\mathcal{C})| - |N_L(\mathcal{F})| \geq t(N_1 - M_1) + \frac{2}{3}(N_2 - M_2) + (N_3 - M_3)$.*

Proof. First, note that $|N_L(\mathcal{C}|A \cup \mathcal{C}|B \cup \mathcal{C}|C)| - |N_L(\mathcal{F}|A \cup \mathcal{F}|B \cup \mathcal{F}|C)| \geq t(|\mathcal{C}|A| - |\mathcal{F}|A| + |\mathcal{C}|B| - |\mathcal{F}|B| + |\mathcal{C}|C| - |\mathcal{F}|C|) = t(N_1 - M_1)$. Next, according to the algorithm in Fig. 2, three networks $Network_A$, $Network_B$, and $Network_C$ are generated. Every triplet t in X_2 and Y_2 should be consistent with two of these three networks. In addition, all triplets

in X_3 and Y_3 are consistent with all three networks. So, N_L , which is $Network_{\mathcal{X}}$ for some $\mathcal{X} \in \{A, B, C\}$, must have $|N_L(X_2 \cup X_3)| - |N_L(Y_2 \cup Y_3)| \geq \frac{2}{3}(N_2 - M_2) + N_3 - M_3$. Thus, in total, $|N_L(\mathcal{C})| - |N_L(\mathcal{F})| \geq t(N_1 - M_1) + \frac{2}{3}(N_2 - M_2) + (N_3 - M_3)$. \square

Given the above lemmas, the next lemma follows.

Lemma 10. $|N_L(\mathcal{C})| - |N_L(\mathcal{F})| \geq \frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|)$.

Proof. The lemma is proved by induction on $|L|$ where $L = \Lambda(\mathcal{C})$. **Base case** ($|L| \leq 3$): If $|L| \leq 3$, we can give a network N that has $|N(\mathcal{C}) - N(\mathcal{F})| \geq \frac{2}{3}(|\mathcal{C}| - |\mathcal{F}|)$. Thus, $|N(\mathcal{C})| - |N(\mathcal{F})| \geq \frac{5}{12}|\mathcal{C}|$. **Inductive case** ($|L| > 3$): Step 1 of Algorithm *Approximate* partitions L into 3 subsets A, B, C and Step 2 computes the three networks K_A, K_B, K_C . By the induction assumption, $|K_X(\mathcal{C}|X)| - |K_X(\mathcal{F}|X)| \geq \frac{5}{12}(|\mathcal{C}|X| - |\mathcal{F}|X|)$ for $X \in \{A, B, C\}$. By Lemmas 8 and 9, $|N_L(\mathcal{C})| - |N_L(\mathcal{F})| \geq \frac{5}{12}(N_1 - M_1) + \frac{2}{3}(N_2 - M_2) + N_3 - M_3 \geq \frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|)$. \square

Finally, the next two lemmas are devoted to analyzing the time complexity.

Lemma 11. Algorithm *LeafPartition* runs in $O(|L||\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$ time.

Proof. Steps 1 can be easily implemented in $O(|L|)$ time. As $score(A, B, C)$ is increased by at least 1 for every iteration and $score(A, B, C) \leq 12|\mathcal{C}|$, the while loop in Step 2 executes for at most $12|\mathcal{C}|$ times and Step 3 is also executed for at most $12|\mathcal{C}|$ times. Each iteration of the while loop needs to compute $O(|L|)$ scores and each score can be computed in $O(|\mathcal{C}| + |\mathcal{F}|)$ time. Thus, Step 2 takes $O(|L||\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$ time. Similarly, Step 3 also takes a total of $O(|L||\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$ time throughout the algorithm. \square

Lemma 12. Algorithm *MMTG*(\mathcal{C}, \mathcal{F}) runs in $O(|L|^2|\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$ time.

Proof. Let $t(\mathcal{C}, \mathcal{F})$ be the running time of *MMTG*(\mathcal{C}, \mathcal{F}) and $f(\mathcal{C}, \mathcal{F})$ be the running time of *LeafPartition*(\mathcal{C}, \mathcal{F}). We have $t(\mathcal{C}, \mathcal{F}) = f(\mathcal{C}, \mathcal{F}) + t(\mathcal{C}|A, \mathcal{F}|A) + t(\mathcal{C}|B, \mathcal{F}|B) + t(\mathcal{C}|C, \mathcal{F}|C)$. By Lemma 11, $f(\mathcal{C}, \mathcal{F}) = O(|L||\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$. By solving the recursive equation, $t(\mathcal{C}, \mathcal{F}) = O(|L|^2|\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$. \square

In conclusion, we have the following theorem.

Theorem 5. Given two sets \mathcal{C} and \mathcal{F} of rooted triplets, a galled network N such that $|N(\mathcal{C})| - |N(\mathcal{F})|$ is at least $\frac{5}{12}(|\mathcal{C}| - |\mathcal{F}|)$ can be constructed in $O(|L|^2|\mathcal{C}|(|\mathcal{C}| + |\mathcal{F}|))$ time.

5. Concluding remarks

In this paper, we have investigated the polynomial-time computability of MT and several of its variants, and proposed some new exact and approximation algorithms. In the future, we plan to further improve the performance of the approximation algorithms and the time complexity of our algorithms.

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 15th Annual ACM Symposium on the Theory of Computing (STOC’83)*, pages 1–9, 1983.
3. V. Berry and F. Nicolas. Maximum agreement and compatible supertrees. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM 2004)*, volume 3109 of *LNCS*, pages 205–219. Springer-Verlag, 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 4th Workshop on Algorithms in Bioinformatics (WABI 2004)*, volume 3240 of *LNCS*, pages 388–399. Springer-Verlag, 2004.
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 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 349–358, 2005.
16. J. Jansson and W.-K. Sung. Inferring a level-1 phylogenetic network from a dense set of rooted triplets. In *Proceedings of the 10th 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 9th 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 8th 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.