Algorithms for Combining Rooted Triplets into a Galled Phylogenetic Network

Jesper Jansson* Nguyen Bao Nguyen* Wing-Kin Sung*,†

Abstract

This paper considers the problem of determining whether a given set $\mathcal{T}$ of rooted triplets can be merged without conflicts into a galled phylogenetic network, and if so, constructing such a network. When the input $\mathcal{T}$ is dense, we solve the problem in $O(|\mathcal{T}|)$ time, which is optimal since the size of the input is $\Theta(|\mathcal{T}|)$. In comparison, the previously fastest algorithm for this problem runs in $O(|\mathcal{T}|^2)$ time. Next, we prove that the problem becomes NP-hard if extended to non-dense inputs, even for the special case of simple phylogenetic networks. We also show that for every positive integer n, there exists some set $\mathcal{T}$ of rooted triplets on n leaves such that any galled network can be consistent with at most $0.4883 \cdot |\mathcal{T}|$ of the rooted triplets in $\mathcal{T}$. On the other hand, we provide a polynomial-time approximation algorithm that always outputs a galled network consistent with at least a factor of $\frac{5}{12}$ (> 0.4166) of the rooted triplets in $\mathcal{T}$.

1 Introduction

A *rooted triplet* is a binary, rooted, unordered tree with three distinctly labeled leaves. Aho *et al.* [1] introduced the problem of determining whether a given set of rooted triplets can be combined without conflicts into a distinctly leaf-labeled tree which contains each of the given rooted triplets as an induced subtree, and if so, returning one. The original motivation for this problem came from an application in the theory of relational databases (see [1] for details), but it has later been further studied and generalized because of its applications to phylogenetic tree construction [2, 6, 7, 9, 12, 13, 16, 19, 20, 22, 24]. Here, we study an extension of the problem in which the objective is to determine if a given set $\mathcal{T}$ of rooted triplets can be merged into a more complex structure known as a *galled phylogenetic network*.

A *phylogenetic network* is a type of distinctly leaf-labeled, directed acyclic graph that can be used to model non-treelike evolution. A number of methods for inferring phylogenetic networks under various assumptions and using different kinds of data have recently been proposed [8, 11, 14, 18, 21, 23]. A *galled phylogenetic network* (or *galled network* for short) is an important, biologically motivated structural restriction of a phylogenetic network [8, 18, 23] in which all cycles in the underlying undirected graph are node-disjoint[1].

We present several new results for the problem of inferring a galled network consistent with a given set $\mathcal{T}$ of rooted triplets. Denote the set of leaf labels in $\mathcal{T}$ by L. If $\mathcal{T}$ contains at least one rooted triplet for each cardinality three subset of L, then $\mathcal{T}$ is called *dense*. We first give an exact algorithm named *FastGalledNetwork* for dense inputs whose running time is $O(|\mathcal{T}|)$. In comparison, the previously fastest known algorithm for this case runs in $O(|\mathcal{T}|^2)$ time [14]. Since the size of the input is $\Theta(|\mathcal{T}|)$ when $\mathcal{T}$ is dense and any algorithm that solves the problem must look at the entire input, the asymptotic running time of our new algorithm is optimal. The improvement in running time is due to two observations: Firstly, that the so-called SN-sets employed in [14] do not have to be explicitly computed but can be represented using a tree (the SN-tree) which we can construct in $O(|\mathcal{T}|)$ time, and secondly, that the SN-tree can be expanded into a galled network consistent with $\mathcal{T}$ (if one exists) in $O(|\mathcal{T}|)$ time by replacing each internal node of degree three or higher with a special kind of galled network found by applying an algorithm called *SimpleNetworks*.

Next, we show that the problem becomes NP-hard when $\mathcal{T}$ is not required to be dense. Finally, we consider approximation algorithms. We present an $O(|L| \cdot |\mathcal{T}|^3)$-time algorithm that always outputs a galled network consistent with at least a factor of $\frac{5}{12}$ (> 0.4166) of the rooted triplets in $\mathcal{T}$, for any $\mathcal{T}$. (Our approximation algorithm can also be applied in the dense case when the

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

†Also affiliated with Genome Institute of Singapore, 60 Biopolis Street, Genome, Singapore 138672.

[1]Without the node-disjoint constraint, our problem becomes trivial to solve since then a solution always exists, and furthermore, can be obtained in polynomial time using a simple sorting network-based construction [14].

input cannot be combined into a galled network without conflicts.) On the negative side, we show that there exist inputs for which any galled network can be consistent with at most a factor of 0.4883 of the rooted triplets. It is interesting to note that for *trees*, the corresponding bounds are known to be tight [7]: that is, there is a polynomial-time approximation algorithm which always constructs a tree consistent with at least $\frac{1}{3} \cdot |\mathcal{T}|$ of the rooted triplets in $\mathcal{T}$, and there exist some inputs for which no tree can achieve a factor higher than $\frac{1}{3} \cdot |\mathcal{T}|$.

1.1 Definitions A *phylogenetic tree* is a binary, rooted, unordered tree whose leaves are distinctly labeled. A *phylogenetic network* is a generalization of a phylogenetic tree formally defined as a rooted, connected, directed acyclic graph in which: (1) exactly one node has indegree 0 (the *root*), and all other nodes have indegree 1 or 2; (2) all nodes with indegree 2 (referred to as *hybrid nodes*) have outdegree 1, and all other nodes have outdegree 0 or 2; and (3) 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* (*galled network*, for short) if all cycles in $\mathcal{U}(N)$ are node-disjoint. Galled networks are also known in the literature as *topologies with independent recombination events* [23], *galled-trees* [8], *gt-networks* [18], and *level-1 phylogenetic networks* [4, 14].

A phylogenetic tree with exactly 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 $(\{x, y\}, z)$. For any phylogenetic network N, 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 said to be *consistent* with N if every rooted triplet in $\mathcal{T}$ is consistent with N.

Denote the set of leaves in any phylogenetic network N by $\Lambda(N)$, and for any set $\mathcal{T}$ of rooted triplets, define $\Lambda(\mathcal{T}) = \bigcup_{t_i \in \mathcal{T}} \Lambda(t_i)$. A set $\mathcal{T}$ of rooted triplets is *dense* if for each $\{x, y, z\} \subseteq \Lambda(\mathcal{T})$, at least one of $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ belongs to $\mathcal{T}$. If $\mathcal{T}$ is dense then $|\mathcal{T}| = \Theta(|\Lambda(\mathcal{T})|^3)$. Furthermore, for any set $\mathcal{T}$ of rooted triplets and $L' \subseteq \Lambda(\mathcal{T})$, define $\mathcal{T} \,|\, L'$ as the subset of $\mathcal{T}$ consisting of all rooted triplets t with $\Lambda(t) \subseteq L'$. The problem we consider here is: Given a set $\mathcal{T}$ of rooted triplets, output a galled network N with $\Lambda(N) = \Lambda(\mathcal{T})$ such that N and $\mathcal{T}$ are consistent, if such a network exists; otherwise, output *null*. See Figure 1 for an example. Throughout this paper, we write $L = \Lambda(\mathcal{T})$ and $n = |L|$.

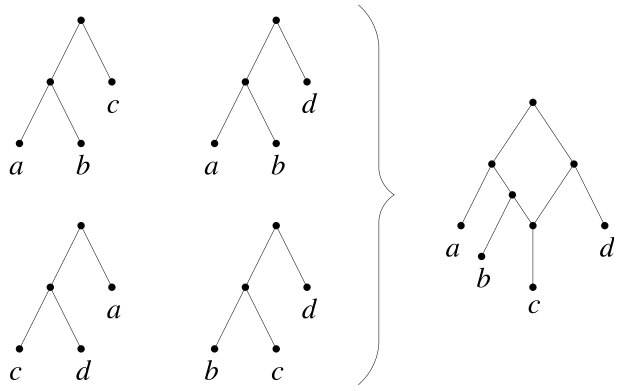

Figure 1: A dense set $\mathcal{T}$ of rooted triplets with leaf set $\{a, b, c, d\}$ and a galled phylogenetic network which is consistent with $\mathcal{T}$. Note that this solution is not unique.

To describe our algorithms, we need the following additional terminology. Let N be a phylogenetic network. We call nodes with indegree 2 *hybrid nodes* and their parent edges *hybrid edges*. 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*, and any path starting at a child of s and ending at a parent of h is a *clipped merge path of h*. From the above, it follows 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.

Let N be a galled network. 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*. N is called a *simple phylogenetic network* (or *simple network*) if N has exactly one hybrid node h, the root node of N is the split node of h, and every side network of N is a leaf. For example, the galled network on the right in Figure 1 is a simple network.

1.2 Applications Phylogenetic networks are used by scientists to describe evolutionary relationships that do not fit the traditional models in which evolution is assumed to be treelike. Evolutionary events such as horizontal gene transfer or hybrid speciation (often referred to as *recombination events*) cannot be adequately represented in a single tree [8, 18, 21, 23] but can be modeled in a phylogenetic network as internal nodes having more than one parent. Galled networks are an important type of phylogenetic networks which have attracted special attention in the literature [4, 8, 18, 23] due to their biological significance (see [8]) and their simple, almost treelike, structure. When the number of recombination events is limited and most of them have occurred recently, a galled network may suffice to accurately describe the evolutionary process under study [8].

A challenge in the field of phylogenetics is to develop efficient and reliable methods for constructing and comparing phylogenetic networks. For example, to construct a meaningful phylogenetic network for a large subset of the human population (which may subsequently be used to help locate regions in the genome associated with some observable trait indicating a particular disease) in the future, efficient algorithms are crucial because the input can be expected to be very large. The motivation behind the rooted triplet approach taken in this paper is that a highly accurate tree for each cardinality three subset of the leaf set can be obtained through maximum likelihood-based methods such as [3] or Sibley-Ahlquist-style DNA-DNA hybridization experiments (see [16]). Hence, the algorithms presented in [14] and in here can be used as the merging step in a divide-and-conquer approach for constructing phylogenetic networks analogous to the quartet method paradigm for inferring unrooted phylogenetic trees [15, 17] and other supertree methods (see [9, 20] and the references therein). We consider dense input sets in particular since this case can be solved in polynomial time.

1.3 Related work Aho *et al.* [1] gave 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 such a tree. Henzinger *et al.* [9] improved its running time to $\min\{O(|\mathcal{T}| \cdot n^{0.5}), O(|\mathcal{T}| + n^2 \log n)\}$; replacing the deterministic algorithm for dynamic graph connectivity employed by Henzinger *et al.* yields a running time of $\min\{O(|\mathcal{T}| \cdot \log^2 n), O(|\mathcal{T}| + n^2 \log n)\}$ [10, 13]. Gąsieniec *et al.* [6] studied a variant of the problem for *ordered* trees. Ng and Wormald [20] considered the problem of constructing *all* rooted, unordered trees distinctly leaf-labeled by $\Lambda(\mathcal{T})$ that are consistent with $\mathcal{T}$.

If two or more of the rooted triplets are in conflict, i.e., contain contradicting branching information, the algorithm of Aho *et al.* returns a null tree. However, this is not very practical in certain applications. For example, in the context of constructing a phylogenetic tree from rooted triplets, some errors may occur in the input when the rooted triplets are based on data obtained experimentally, yet a non-null tree is still required. In this case, one can try to construct a tree consistent with the maximum number of rooted triplets in the input [2, 6, 7, 12, 24], or a tree with as many leaves from $\Lambda(\mathcal{T})$ as possible which is consistent with all input rooted triplets involving these leaves only [13]. Although the former problem is NP-hard [2, 12, 24], Gąsieniec *et al.* [7] showed that it has a polynomial-time approximation algorithm that outputs a distinctly leaf-labeled tree consistent with at least $\frac{1}{3}$ of the given rooted triplets, which is a tight bound in the sense that there exist inputs $\mathcal{T}$ such that any distinctly leaf-labeled tree can be consistent with at most $\frac{1}{3}$ of the rooted triplets in $\mathcal{T}$.

The problem studied in this paper was introduced in [14]. The main result of [14] is an exact $O(|\mathcal{T}|^2)$-time algorithm for the dense case. [14] also showed that if no restrictions are placed on the structure of the output phylogenetic network (i.e., if non-galled networks are allowed) then the problem always has a solution which can easily be obtained from any given sorting network for n elements. Nakhleh *et al.* [18] gave an $O(n^2)$-time algorithm for the related problem of combining two given phylogenetic trees T_1 and T_2 with identical leaf sets into a galled network containing both T_1 and T_2 as induced subtrees, where n is the number of leaves (in fact, this is equivalent to inferring a galled network consistent with the set of all rooted triplets which are induced subtrees of T_1 or T_2). They also studied the case where T_1 and T_2 may contain errors but only one hybrid node is allowed. Huson *et al.* [11] considered a similar problem for constructing an *unrooted* phylogenetic network from a set of unrooted, distinctly leaf-labeled trees.

1.4 Organization of the paper In Section 2, we present a new algorithm called *SimpleNetworks* for computing all simple phylogenetic networks consistent with a given dense set $\mathcal{T}$ of rooted triplets in $O(n^3)$ time. This algorithm is used by our main algorithm *FastGalledNetwork* in Section 3 to construct a galled network consistent with a given dense set $\mathcal{T}$ of rooted triplets, if one exists, in optimal $O(n^3)$ time. In Section 4, we prove that the problem becomes NP-hard if we remove the requirement that $\mathcal{T}$ forms a dense set. Next, in Section 5.1, we show that for every positive integer n, there exists some set $\mathcal{T}$ of rooted triplets with $|\Lambda(T)| = n$ such that any galled network can be consistent with at most $0.4883 \cdot |\mathcal{T}|$ of the rooted triplets in $\mathcal{T}$. On the other hand, we give an $O(n \cdot |\mathcal{T}|^3)$-time algorithm in Section 5.2 that constructs a galled network guaranteed to be consistent with at least a factor of $\frac{5}{12}$ (> 0.4166) of the rooted triplets in $\mathcal{T}$ for any input $\mathcal{T}$.

2 Constructing all simple phylogenetic networks when $\mathcal{T}$ is dense

In this section, we describe an algorithm called *SimpleNetworks* for inferring all simple phylogenetic networks consistent with a given dense set $\mathcal{T}$ of rooted triplets in $O(n^3)$ time, where $L = \Lambda(\mathcal{T})$ and $n = |L|$. This algorithm is later used by our main algorithm in Section 3. Below, for any $L' \subseteq L$, $\mathcal{G}(L')$ denotes *the auxiliary graph for* L' (originally defined by Aho *et al.* [1]), which is the undirected graph with vertex set L' and edge set $E(L')$ where for each $(\{i,j\},k) \in \mathcal{T} \,|\, L'$, the edge $\{i,j\}$ is included in $E(L')$. *SimpleNetworks* assumes that $n \geq 3$, $\mathcal{T}$ is dense, and $\mathcal{G}(L)$ is connected.

For any simple network N, define $A(N)$ and $B(N)$ to be the sets of leaves attached to the two clipped merge paths in N, where we require without loss of generality that $A(N)$ is nonempty. If both $A(N)$ and $B(N)$ are nonempty then N is called *non-skew*; if $A(N)$ is nonempty and $B(N)$ is empty then N is called *skew*. Denote the leaf attached to the hybrid node of N by $h(N)$.

Algorithm *SimpleNetworks* calls two procedures named *Non-SkewSimpleNetworks* and *SkewSimpleNetworks* that find all valid non-skew simple networks and all valid skew simple networks, respectively. Then, it returns their union. In the next two subsections, we show how to implement each of these two procedures to run in $O(n^3)$ time. We thus obtain Theorem 2.1.

Theorem 2.1. *The set of all simple networks consistent with a given dense set of rooted triplets and leaf-labeled by L can be constructed in $O(n^3)$ time.*

The next two lemmas are used in Sections 2.1 and 2.2. A *caterpillar tree* is a rooted tree such that every internal node has at most one child which is not a leaf (see, e.g. [2]).

Lemma 2.1. *Suppose N is a simple phylogenetic network that is consistent with a set $\mathcal{T}$ of rooted triplets. Let N' be the graph obtained from N by deleting the root node of N, the hybrid node of N, and $h(N)$ together with all their incident edges, and then, for every node with outdegree 1 and indegree less than 2, contracting its outgoing edge. If N is non-skew then N' consists of two binary caterpillar trees which are consistent with $\mathcal{T} \,|\, A(N)$ and $\mathcal{T} \,|\, B(N)$, respectively; if N is skew then N' is a binary caterpillar tree which is consistent with $\mathcal{T} \,|\, A(N)$.*

Lemma 2.2. [14] *Let $\mathcal{T}$ be a dense set of rooted triplets and let L be the leaf set of $\mathcal{T}$. There is at most one rooted, unordered tree distinctly leaf-labeled by L which is consistent with $\mathcal{T}$. Furthermore, if such a tree exists then it must be binary.*

2.1 Constructing all non-skew simple phylogenetic networks Let U be an undirected, connected graph. Any partition (X, Y, Z) of the vertex set of U is called a *non-skew leaf partition in U* if $|X| \geq 1$, $|Y| = 1$, $|Z| \geq 1$, and in U the following holds: (1) X and Z form two cliques; and (2) there is no edge between a vertex in X and a vertex in Z. Any two non-skew leaf partitions of the form (X, Y, Z) and (Z, Y, X) are considered to be equivalent.

Lemma 2.3. *Let $\mathcal{T}$ be a dense set of rooted triplets and let L be the leaf set of $\mathcal{T}$. If N is a non-skew simple phylogenetic network with leaf set L that is consistent with $\mathcal{T}$ then $(A(N),\, h(N),\, B(N))$ forms a non-skew leaf partition in $\mathcal{G}(L)$.*

By Lemmas 2.1 and 2.3, if N is a non-skew simple network with leaf set L that is consistent with $\mathcal{T}$ then $(A(N),\, h(N),\, B(N))$ forms a non-skew leaf partition in $\mathcal{G}(L)$ and $\mathcal{T} \,|\, A(N)$ and $\mathcal{T} \,|\, B(N)$ are consistent with two binary caterpillar trees. Algorithm *Non-SkewSimpleNetworks*, shown in Figure 2, uses these implications to construct all non-skew simple networks with leaf set L that are consistent with $\mathcal{T}$. The algorithm enumerates all non-skew leaf partitions in $\mathcal{G}(L)$, and for each such leaf partition P, tries to build binary caterpillar trees for subsets of L induced by P (Lemma 2.2 ensures that for any dense subset $\mathcal{T}'$ of $\mathcal{T}$, if $\mathcal{T}'$ is consistent with a caterpillar tree then it is uniquely determined, so the algorithm of Aho *et al.* [1] can find it) and if successful, then combines the caterpillar trees in accordance with Lemma 2.1 to obtain all possible valid simple networks. Lemmas 2.1 and 2.3 guarantee that this approach will discover every valid simple network. However, it may also yield some simple networks which are not consistent with $\mathcal{T}$; hence, before including any constructed network N in the final solution set $\mathcal{N}_1$, *Non-SkewSimpleNetworks* verifies if N is consistent with $\mathcal{T}$.

For any $L' \subseteq L$ with $|L'| \geq 3$, *BuildTree*$(\mathcal{T} \,|\, L')$ refers to the fast implementation of the algorithm of Aho *et al.* applied to $\mathcal{T} \,|\, L'$ (we may assume it returns *null* if it fails). For $|L'| < 3$, the set $\mathcal{T} \,|\, L'$ is empty and we simply let *BuildTree*$(\mathcal{T} \,|\, L')$ return a tree with the leaves in L'. The running time of *BuildTree*$(\mathcal{T} \,|\, L')$ is $\min\{O(|\mathcal{T}| \cdot \log^2 n),\, O(|\mathcal{T}| + n^2 \log n)\}$ (see Section 1.3).

The next lemma is easy to prove after observing that for any two non-skew leaf partitions $(X, \{h\}, Z)$ and $(X', \{h'\}, Z')$ in an undirected connected graph, h and h' must be neighbors.

Lemma 2.4. *Any undirected, connected graph U has at most two non-skew leaf partitions.*

Theorem 2.2. *The time complexity of Algorithm Non-SkewSimpleNetworks is $O(n^3)$.*

```
Algorithm Non-SkewSimpleNetworks
Input: A dense set $\mathcal{T}$ of rooted triplets such that $\mathcal{G}(L)$ consists of one connected component.
Output: The set of all non-skew simple networks with leaf set $L$ which are consistent with $\mathcal{T}$.
1   Set $\mathcal{N}_1 = \emptyset$.
2   Construct $\mathcal{G}(L)$ and compute all non-skew leaf partitions in $\mathcal{G}(L)$.
3   for each non-skew leaf partition $(X, Y, Z)$ in $\mathcal{G}(L)$ do
3.1     Let $T_X = BuildTree(\mathcal{T} \mid X)$ and $T_Z = BuildTree(\mathcal{T} \mid Z)$.
3.2     if $T_X$ and $T_Z$ are binary caterpillar trees then
            Make the roots of $T_X$ and $T_Z$ children of a new root node, create a new hybrid node $H$
            with a child labeled by the leaf in $Y$, and construct at most four non-skew simple networks
            by attaching $H$ to one of $T_X$'s bottommost leaves' parent edges and one of $T_Z$'s
            bottommost leaves' parent edges in all possible ways. For each obtained network $N$, if $N$
            is consistent with $\mathcal{T}$ then $\mathcal{N}_1 = \mathcal{N}_1 \cup \{N\}$.
    endfor
4   return $\mathcal{N}_1$.
End Non-SkewSimpleNetworks
```

Figure 2: Constructing all non-skew simple networks.

```
Algorithm SkewSimpleNetworks
Input: A dense set $\mathcal{T}$ of rooted triplets such that $\mathcal{G}(L)$ consists of one connected component.
Output: The set of all skew simple networks with leaf set $L$ which are consistent with $\mathcal{T}$.
1   Set $\mathcal{N}_2 = \emptyset$.
2   Construct $\mathcal{D}$.
3   for every $x \in L$ do
3.1     Let $Q = BuildCaterpillar(\mathcal{D} \mid L')$, where $L' = L \setminus \{x\}$.
3.2     if $Q \neq null$ then
            Make the root of $Q$ a child of a new root node $r$, create a new hybrid node $H$ with a
            child labeled by $x$, add an edge from $r$ to $H$, and construct two skew simple networks by
            attaching $H$ to each one of $Q$'s two bottommost edges. For each obtained network $N$, if
            $N$ is consistent with all rooted triplets in $\mathcal{T}$ involving $x$ then $\mathcal{N}_2 = \mathcal{N}_2 \cup \{N\}$.
    endfor
4   return $\mathcal{N}_2$.
End SkewSimpleNetworks
```

Figure 3: Constructing all skew simple networks.

2.2 Constructing all skew simple phylogenetic networks To obtain all skew simple networks with leaf set L consistent with $\mathcal{T}$, Algorithm *SkewSimpleNetworks* in Figure 3 tries all ways to remove one leaf x from L and construct a binary caterpillar tree consistent with all rooted triplets not involving x using a procedure named *BuildCaterpillar*. For each such caterpillar Q, it forms two candidate skew simple networks by letting the root of Q be a child of a new split node with a hybrid node H such that H has a child labeled by x and H is attached to one of Q's two bottommost edges. (By Lemma 2.1, every skew simple network with leaf set L that is consistent with $\mathcal{T}$ must have this structure.) Then, each candidate skew simple network is checked to see if it is consistent with all rooted triplets in $\mathcal{T}$ involving x (by the above, it is always consistent with the rest); if yes then it is included in the solution set $\mathcal{N}_2$.

The procedure *BuildCaterpillar* uses a graph $\mathcal{D}$, defined as follows. Given a set $\mathcal{T}$ of rooted triplets with leaf set L, let $\mathcal{D}$ be the directed graph with vertex set L such that there is a directed edge (x, y) if and only if $\mathcal{T}$ contains at least one rooted triplet of the form $(\{y, z\}, x)$, where $z \in L$. For any $L' \subseteq L$, let $\mathcal{D} \mid L'$ be the subgraph of $\mathcal{D}$ in which all vertices not in L' and their incident edges have been deleted. $\mathcal{D} \mid L'$ is acyclic if and only if there exists a binary caterpillar tree consistent with $\mathcal{T} \mid L'$.

For any $L' \subseteq L$, *BuildCaterpillar*$(\mathcal{D} \mid L')$ returns a binary caterpillar tree with leaf set L' which is consistent with $\mathcal{T} \mid L'$ if such a tree exists, and *null* otherwise, by the following method. If $\mathcal{D} \mid L'$ has a cycle then return *null*. Else, do a topological sort of $\mathcal{D} \mid L'$ to find a linear ordering $\mathcal{O}$ of L' and return a binary caterpillar tree whose leaves are labeled in order of increasing distance from the root according to $\mathcal{O}$. Since $\mathcal{T}$ is dense, $\mathcal{O}$ is uniquely determined except for its last two elements which may be interchanged arbitrarily.

THEOREM 2.3. *The time complexity of Algorithm SkewSimpleNetworks is $O(n^3)$.*

3 An exact algorithm for inferring a galled phylogenetic network from a dense set of rooted triplets with optimal running time

Here, we present our algorithm *FastGalledNetwork* for constructing a galled network consistent with a given dense set $\mathcal{T}$ of rooted triplets, if such a network exists. Its running time is $O(n^3)$, where $n = |L|$ and L denotes the leaf set of $\mathcal{T}$, which is optimal since the size of the input is $\Theta(n^3)$ when $\mathcal{T}$ is dense. In Section 3.1, we give an algorithm named *ComputeSNTree* which computes the so-called SN-tree for $\mathcal{T}$ in $O(n^3)$ time. Then, in Section 3.2, we describe *FastGalledNetwork*. It uses *ComputeSNTree* as well as *SimpleNetworks* from Section 2 to construct a galled network consistent with $\mathcal{T}$ (if one exists) from the SN-tree for $\mathcal{T}$. The key observation is that each internal node of degree three or higher in the SN-tree corresponds to a simple network whose structure can be inferred by using *SimpleNetworks*.

3.1 Computing the SN-tree For any $X \subseteq L$, the set $SN(X)$ is defined recursively as $SN(X \cup \{c\})$ if there exist some $x, x' \in X$ and $c \in L \setminus X$ such that $(\{x, c\}, x') \in \mathcal{T}$, and as X otherwise. SN-sets were introduced in [14]. When $\mathcal{T}$ is dense, the SN-sets satisfy the following important property.

Lemma 3.1. [14] *If $\mathcal{T}$ is dense then for any $A, B \subseteq L$, $SN(A) \cap SN(B)$ equals $\emptyset$, $SN(A)$, or $SN(B)$.*

[14] showed how to compute $SN(\{a, b\})$ for any $a, b \in L$ (where a may be equal to b) in $O(n^3)$ time; that approach therefore takes $O(n^5)$ time to compute $SN(\{a, b\})$ for *all* $a, b \in L$. This section presents a faster method for implicitly computing all SN-sets of this form when $\mathcal{T}$ is dense which only requires $O(n^3)$ time. The algorithm (*ComputeSNTree*) is listed in Figure 4. Given a dense $\mathcal{T}$, it builds a rooted tree called *the SN-tree for $\mathcal{T}$* which encodes all SN-sets so that $SN(X)$ for any $X \subseteq L$ can be retrieved efficiently.

ComputeSNTree first constructs a directed graph $G_{\mathcal{T}}$ with vertex set $V(G_{\mathcal{T}})$ and edge set $E(G_{\mathcal{T}})$. $V(G_{\mathcal{T}})$ is defined as $\{v_{\{a,b\}} \mid a, b \in L\}$, where $v_{\{a,a\}}$ for any $a \in L$ is denoted by $v_{\{a\}}$ for short, and $E(G_{\mathcal{T}})$ is $\{(v_{\{a,c\}}, v_{\{a,b\}}), (v_{\{a,c\}}, v_{\{b,c\}}), (v_{\{b,c\}}, v_{\{a,b\}}), (v_{\{b,c\}}, v_{\{a,c\}}) \mid (\{a, b\}, c) \in \mathcal{T}\} \bigcup \{(v_{\{a,b\}}, v_{\{a\}}), (v_{\{a,b\}}, v_{\{b\}}) \mid a, b \in L\}$. Note that $|V(G_{\mathcal{T}})| = O(n^2)$ and $|E(G_{\mathcal{T}})| = O(n^3)$.

Lemma 3.2. *Suppose $\mathcal{T}$ is dense. For every $a, b \in L$, we have $c \in SN(\{a, b\})$ if and only if there exists a directed path from $v_{\{a,b\}}$ to $v_{\{c\}}$ in $G_{\mathcal{T}}$.*

Corollary 3.1. *For any two nodes $v_{\{a,b\}}$ and $v_{\{c,d\}}$ on a directed cycle in $G_{\mathcal{T}}$, $SN(\{a, b\}) = SN(\{c, d\})$.*

By the above, computing $SN(\{a, b\})$ is equivalent to finding all nodes of the form $v_{\{c\}}$ reachable from $v_{\{a,b\}}$. Let the set of all strongly connected components of $G_{\mathcal{T}}$ be $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$. By Corollary 3.1, $SN(\{w, x\}) = SN(\{y, z\})$ if $v_{\{w,x\}}$ and $v_{\{y,z\}}$ are in the same C_i. So, we define $SN(C_i)$ as $SN(\{w, x\})$ for any $v_{\{w,x\}} \in C_i$. The set $\mathcal{C}$ has the following properties.

Lemma 3.3. *For every $c \in L$, $\{v_{\{c\}}\} \in \mathcal{C}$. Also, $|\mathcal{C}| = O(n)$, and for every $i \neq j$, $SN(C_i) \neq SN(C_j)$.*

Let $G'_{\mathcal{T}}$ be the directed graph with $V(G'_{\mathcal{T}}) = \mathcal{C}$ and $E(G'_{\mathcal{T}}) = \{(C_i, C_j) \mid$ there exists some $(v_{\{w,x\}}, v_{\{y,z\}}) \in E(G_{\mathcal{T}})$ where $v_{\{w,x\}} \in C_i$ and $v_{\{y,z\}} \in C_j\}$. Note that $G'_{\mathcal{T}}$ is a directed acyclic graph. From $G'_{\mathcal{T}}$, we construct a graph $R_{\mathcal{T}}$ with $V(R_{\mathcal{T}}) = \mathcal{C}$ and $E(R_{\mathcal{T}}) = \{(C_i, C_j) \in E(G'_{\mathcal{T}}) \mid$ there exists no path of length at least 2 from C_i to C_j in $G'_{\mathcal{T}}\}$. Finally, for every vertex in $R_{\mathcal{T}}$ with outdegree 1, contract its outgoing edge. The next lemma implies that $R_{\mathcal{T}}$ is a tree.

```
Algorithm  ComputeSNTree
Input:   A dense set $\mathcal{T}$ of rooted triplets.
Output:  The $SN$-tree $R_{\mathcal{T}}$ for $\mathcal{T}$.

1  Build the directed graph $G_{\mathcal{T}}$ and compute the set $\mathcal{C}$
   of strongly connected components of $G_{\mathcal{T}}$.
2  Build $G'_{\mathcal{T}}$ for $\mathcal{T}$, build the $SN$-tree $R_{\mathcal{T}}$, and return $R_{\mathcal{T}}$.
End  ComputeSNTree
```

Figure 4: Computing the SN-tree for a dense set $\mathcal{T}$.

Lemma 3.4. *$R_{\mathcal{T}}$ is a tree with n leaves and no nodes with outdegree 1. Its set of leaves is $\{\{v_{\{c\}}\} \mid c \in L\}$.*

In the rest of the paper, $R_{\mathcal{T}}$ is called the *SN-tree for $\mathcal{T}$*. This is because $SN(\{a, b\})$ for any $a, b \in L$ can be obtained from $R_{\mathcal{T}}$ using the following lemma.

Lemma 3.5. *Given any $a, b \in L$, let u be the lowest common ancestor of $v_{\{a\}}$ and $v_{\{b\}}$ in $R_{\mathcal{T}}$. Then, $SN(\{a, b\}) = \{c \in L \mid v_{\{c\}}$ is a descendant of u in $R_{\mathcal{T}}\}$.*

We have:

Theorem 3.1. *The time complexity of Algorithm ComputeSNTree is $O(n^3)$.*

3.2 Algorithm *FastGalledNetwork* The main algorithm of this section, Algorithm *FastGalledNetwork*, is listed in Figure 5. Recall that for any node u in a rooted, leaf-labeled tree R, $R[u]$ is the subtree of R rooted at u and $\Lambda(R[u])$ denotes the set of leaves in $R[u]$. Below, $\mathcal{T} \mid u$ is shorthand for the set $\mathcal{T} \mid \Lambda(R[u])$.

In Step 1, *FastGalledNetwork* computes the SN-tree R for $\mathcal{T}$ with Algorithm *ComputeSNTree* from Section 3.1. Then, in Steps 2 and 3, it tries to construct a galled network N_u consistent with all rooted triplets in $\mathcal{T} \mid u$ for each node u in R in bottom-up order. If successful, it returns N_r, where r is the root of R (note that $\mathcal{T} = \mathcal{T} \mid r$); otherwise, it returns *null*. To obtain N_u for any node u in R, *FastGalledNetwork* proceeds as follows. Let q be the degree of u and denote the children of u by $\{u_1, u_2, \ldots, u_q\}$. If $q = 0$ then let N_u be a network consisting of one leaf, labeled by u. If $q = 2$ then form N_u by joining the roots of N_{u_1} and N_{u_2} to a new root node. Otherwise, $q \geq 3$ by Lemma 3.4. In this case, let $\alpha_1, \alpha_2, \ldots, \alpha_q$ be q new symbols not in L, and define a function f as follows. For every

```
Algorithm  FastGalledNetwork
Input:   A dense set $\mathcal{T}$ of rooted triplets.
Output:  A galled network consistent with $\mathcal{T}$, if one
         exists; otherwise, null.
1   Let $R$ = ComputeSNTree($\mathcal{T}$).
2   Define $N_u$ for every leaf $u$ in $R$ to be a single node
    labeled by $u$.
3   for each internal node $u$ in $R$ in bottom-up order do
        /* Construct a galled network $N_u$ for $\Lambda(R[u])$. */
3.1     Denote the set of children of $u$ in $R$ by
        $\{u_1, u_2, \ldots, u_q\}$.
3.2     If $q = 2$, let $N_u$ be a network with a root node
        joined to $N_{u_1}$ and $N_{u_2}$.
3.3     Otherwise ($q \geq 3$), build $\mathcal{T}'$ from $\mathcal{T} \mid u$, compute
        $\mathcal{N}$ = SimpleNetworks($\mathcal{T}'$), and check if $\mathcal{N}$ is
        empty; if yes then return null, else select any
        $N' \in \mathcal{N}$ and form a network $N_u$ by replacing
        each $\alpha_i$ in $N'$ with $N_{u_i}$.
    endfor
4   return $N_r$, where $r$ is the root of $R$.
End  FastGalledNetwork
```

Figure 5: Constructing a galled phylogenetic network consistent with a dense set $\mathcal{T}$ of rooted triplets.

$x \in \Lambda(R[u])$, let $f(x) = \alpha_i$ where $x \in \Lambda(R[u_i])$. Next, define $\mathcal{T}'$ as the set $\{(\{f(x), f(y)\}, f(z)) : (\{x, y\}, z) \in (\mathcal{T} \mid u)$ and $f(x), f(y), f(z)$ all differ$\}$, and apply Algorithm *SimpleNetworks* from Section 2 to $\mathcal{T}'$. If there is a simple phylogenetic network N' consistent with $\mathcal{T}'$ then replace each α_i in N' with N_{u_i} and let N_u be the resulting network; otherwise, terminate and output *null*.

The correctness of this method follows from the next two lemmas.

LEMMA 3.6. *For any node u in R, if $\mathcal{T} \mid u$ is consistent with a galled network with leaf set $\Lambda(R[u])$ and if $q \geq 3$ then there exists a simple network consistent with $\mathcal{T}'$.*

Proof. (Sketch.) Let M be any galled network with leaf set $\Lambda(R[u])$ consistent with $\mathcal{T} \mid u$. First show that if $q \geq 3$ then the root of M is a split node of some hybrid node h, and then that each side network attached to a merge path of h contains leaves from only one $\Lambda(R[u_i])$. Next, observe that by concatenating side networks containing leaves from the same $\Lambda(R[u_i])$, M can be transformed into a galled network M^* consistent with $\mathcal{T} \mid u$ such that each side network M_i^* attached to a merge path of h is bijectively leaf-labeled by one $\Lambda(R[u_i])$. Construct a simple phylogenetic network M' from M^* by replacing each M_i^* by a leaf labeled by α_i. Then, M' is consistent with $\mathcal{T}'$. □

LEMMA 3.7. *Let u be any node in R and suppose each $\mathcal{T} \mid u_i$ is consistent with a galled network N_{u_i}. If $q = 2$ then the galled network obtained by joining N_{u_1} and N_{u_2} to a new root node is consistent with $\mathcal{T} \mid u$. If $q \geq 3$ and $\mathcal{T}'$ is consistent with a simple network N' with leaf set $\{\alpha_1, \alpha_2, \ldots, \alpha_q\}$ then the galled network obtained from N' by replacing each α_i by N_{u_i} is consistent with $\mathcal{T} \mid u$.*

Proof. Analogous to the proof of Lemma 8 in [14] (just substitute $\mathcal{T}$ with $\mathcal{T} \mid u$ and each SN_i with $\Lambda(R[u_i])$). □

THEOREM 3.2. *The time complexity of Algorithm FastGalledNetwork is $O(n^3)$.*

We remark that *FastGalledNetwork* can be modified to return *all* galled networks consistent with $\mathcal{T}$ by utilizing all simple networks computed in Step 3.3. However, this may take exponential time.

4 NP-hardness of the non-dense case

We now prove that the problem of inferring a galled phylogenetic network which is consistent with a given set of $\mathcal{T}$ rooted triplets, if one exists, is NP-hard when $\mathcal{T}$ is not required to be dense. Our proof consists of a polynomial-time reduction from the NP-complete problem Set Splitting (see, e.g., [5]) to the decision version of our problem. We use the same reduction to prove that the closely related problem of inferring a *simple* phylogenetic network which is consistent with a given (non-dense) set of rooted triplets is also NP-hard.

Set Splitting: Given a set $S = \{s_1, s_2, \ldots, s_n\}$ and a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of subsets of S where $|C_j| = 3$ for every $C_j \in \mathcal{C}$, does $(S, \mathcal{C})$ have a set splitting, i.e., can S be partitioned into two disjoint subsets S_1, S_2 such that for every $C_j \in \mathcal{C}$ it holds that C_j is not a subset of S_1 and C_j is not a subset of S_2?

First, we describe the reduction from Set Splitting. Given an instance $(S, \mathcal{C})$, construct a non-dense set $\mathcal{T}$ of rooted triplets having a leaf set L with $L = \{h, x, y\} \cup \{s_i^j \mid s_i \in S,\ 1 \leq j \leq m\}$, where h, x, y, and all s_i^j are new elements not belonging to S. Initially, let $\mathcal{T}$ consist of the two rooted triplets $(\{x, h\}, y)$ and $(\{y, h\}, x)$. Next, for each $C_j \in \mathcal{C}$, write $C_j = \{s_a, s_b, s_c\}$ with $a < b < c$ and include three rooted triplets $(\{s_a^j, h\}, s_b^j)$, $(\{s_b^j, h\}, s_c^j)$, and $(\{s_c^j, h\}, s_a^j)$ in $\mathcal{T}$. Finally, for each $s_i \in S$, add m rooted triplets $(\{s_i^1, s_i^2\}, h)$, $(\{s_i^2, s_i^3\}, h)$, $\ldots$, $(\{s_i^{m-1}, s_i^m\}, h)$, $(\{s_i^m, s_i^1\}, h)$ and $2m$ rooted triplets $(\{s_i^1, h\}, x)$, $(\{y, h\}, s_i^1)$, $(\{s_i^2, h\}, x)$, $(\{y, h\}, s_i^2)$, $\ldots$, $(\{s_i^m, h\}, x)$, $(\{y, h\}, s_i^m)$ to $\mathcal{T}$. (In the reduction, $\mathcal{C}$ is encoded by rooted triplets of the form $(\{s_a^j, h\}, s_b^j)$. The purpose of the other rooted triplets is to force any galled network N consistent with $\mathcal{T}$ to have a special structure; see Lemma 4.1. Then, for each $C_j = \{s_a, s_b, s_c\} \in \mathcal{C}$, at most two of s_a^j, s_b^j, s_c^j can descend from the same clipped merge path from the root in N, inducing a set splitting.)

Lemma 4.1. *Suppose N is a galled network with leaf set L which is consistent with $\mathcal{T}$. Then: (1) the root r of N is a split node; (2) one side network attached to a merge path of r contains h but no other leaves; and (3) h is a descendant of the hybrid node for r.*

Theorem 4.1. *Given any non-dense set $\mathcal{T}$ of rooted triplets, it is NP-hard to determine if there exists a galled network which is consistent with $\mathcal{T}$. It is also NP-hard to determine if there exists a simple network which is consistent with $\mathcal{T}$.*

Proof. (Sketch.) First show that if $(S, \mathcal{C})$ has a set splitting then there exists a simple network consistent with $\mathcal{T}$. Next, use Lemma 4.1 to prove that if there exists a galled network consistent with $\mathcal{T}$ then $(S, \mathcal{C})$ has a set splitting. Since a simple phylogenetic network is always a galled network, the theorem follows. □

5 Approximating the maximum number of consistent rooted triplets

This section studies the problem of constructing a galled network consistent with the maximum number of rooted triplets in $\mathcal{T}$ for any (not necessarily dense) given $\mathcal{T}$. Section 5.2 presents a polynomial-time approximation algorithm for this problem which always outputs a galled network consistent with at least a factor of $\frac{5}{12}$ (> 0.4166) of the rooted triplets in $\mathcal{T}$. On the negative side, Section 5.1 shows that there exist inputs for which any galled network can be consistent with at most a factor of 0.4883 of the rooted triplets in $\mathcal{T}$.

5.1 Inapproximability result Given any positive integer n, fix $\mathcal{T}$ to contain all possible rooted triplets for a leaf set L of size n, that is, $\mathcal{T} = \{(\{a,b\},c), (\{a,c\},b), (\{b,c\},a) \mid a,b,c \in L\}$. For any phylogenetic network N, let $\#N$ denote the number of rooted triplets from $\mathcal{T}$ that are consistent with N.

Lemma 5.1. *Let N be a galled network with $\Lambda(N) = L$. If N contains a non-split node u with two children u_1, u_2 such that u is the root node or a child of a hybrid node, then making u into a split node by removing the edges (u, u_1) and (u, u_2), adding two new nodes v and w, and inserting the edges (u, v), (u, w), (v, u_1), (w, u_2), and (v, w) yields a galled network N' with $\#N' = \#N$.*

Lemma 5.2. *Let N be a galled network with $\Lambda(N) = L$. Suppose N contains a merge path P of a hybrid node h, c is the child of h, $N[u]$ is a side network attached to P, $u \neq c$, and u has two children u_1, u_2. If $|N[u]| \geq |N[c]|$ then N can be transformed to a galled network N' with $\#N' \geq \#N$ by letting $N[u]$ and $N[c]$ trade places. Else, if $|N[u]| \leq |N[c]|$ then N can be transformed to a galled network N' with $\#N' \geq \#N$ by removing $N[u]$ and instead attaching $N[u_1]$ and $N[u_2]$ to P, and removing a hybrid edge in case u was a split node.*

By repeatedly applying Lemmas 5.1 and 5.2, the next lemma concludes that for any fixed n, at least one of the galled networks N for a set of n leaves that maximizes $\#N$ must be a *caterpillar network*. A galled network N is called a caterpillar network if, for every merge path P in N, all side networks attached to P except for the one at the hybrid node are leaves.

Lemma 5.3. *For any galled network N, there is a caterpillar network N' with $\Lambda(N') = \Lambda(N)$ and $\#N' \geq \#N$.*

Now, we are ready to show the bound on the approximation ratio. Let $S(n)$ be the maximum value of $\#N$ taken over all galled networks N with n leaves.

Lemma 5.4. $S(n) = \max_{1 \leq a \leq n}\left\{\binom{a}{3} + 2 \cdot \binom{a}{2} \cdot (n-a) + a \cdot \binom{n-a}{2} + S(n-a)\right\}$.

Proof. (Sketch.) By Lemma 5.3, there is a caterpillar network N that maximizes $\#N$ among all galled networks with n leaves. The recurrence for $S(n)$ counts the maximum number of rooted triplets in $\mathcal{T}$ consistent with a caterpillar network with n leaves. □

Theorem 5.1. *There is no approximation algorithm with approximation ratio larger than* 0.4883.

Proof. Define $T(n) = |\mathcal{T}|$ for any given positive integer n, i.e., $T(n) = 3 \cdot \binom{n}{3}$. Note that the approximation ratio can be at most $\min_{n \in \mathbb{Z}^+} \frac{S(n)}{T(n)}$. By inserting $n = 1000$ into the recurrence in Lemma 5.4, we obtain $S(1000) = 243383298$. Hence, the approximation ratio must be less than or equal to $\frac{S(1000)}{T(1000)} < 0.4883$. □

5.2 A polynomial-time $\frac{5}{12}$-approximation algorithm Given any set $\mathcal{T}$ of rooted triplets, our approximation algorithm called *Approximate* (shown in Figure 6) infers a galled network which is consistent with at least $\frac{5}{12}$ of the rooted triplets in $\mathcal{T}$. We first describe the algorithm and then present the analysis.

Initially, *Approximate* partitions the set of leaves L into three subsets A, B, C so that none of them equals L using an algorithm named *LeafPartition* (described later). Then, for each $X \in \{A, B, C\}$, it recursively infers a galled network K_X by calling *Approximate*$(\mathcal{T}|X)$. 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. Finally, it returns the best network among $Network_A$, $Network_B$, and $Network_C$.

To complete the description of the algorithm, we need to explain how *LeafPartition* divides L into the three subsets A, B, C. We partition L so that a special condition $5N_1 + 8N_2 + 12N_3 \geq 5 \cdot |\mathcal{T}|$ holds, where for $i \in \{0, 1, 2, 3\}$, we define $N_i = |Z_i(A, B, C)|$ and where $Z_i(A, B, C)$ is the set defined as follows:

- $Z_0(A, B, C) = \{(\{x, y\}, z) \in \mathcal{T} \mid x$ and z are in one of the subsets A, B, C and y is in another$\}$;
- $Z_1(A, B, C) = \{(\{x, y\}, z) \in \mathcal{T} \mid x$, y, and z are in one of the subsets $A, B, C\}$;
- $Z_2(A, B, C) = \{(\{x, y\}, z) \in \mathcal{T} \mid x$, y, and z are in three different subsets among $A, B, C\}$;
- $Z_3(A, B, C) = \{(\{x, y\}, z) \in \mathcal{T} \mid x$ and y are in one of the subsets A, B, C and z is in another$\}$.

Note that $Z_0(A, B, C) \cup Z_1(A, B, C) \cup Z_2(A, B, C) \cup Z_3(A, B, C) = \mathcal{T}$. As shown below, any A, B, C which imply $5N_1 + 8N_2 + 12N_3 \geq 5 \cdot |\mathcal{T}|$ guarantee a good approximation ratio. Algorithm *LeafPartition* (also listed in Figure 6) is a greedy algorithm which performs the partitioning. It first divides L into three subsets arbitrarily, and then moves leaves (one at a time) from one subset to another until $score(A, B, C)$ cannot be further improved, where we define $score(A, B, C) = 4N_1 + 7N_2 + 12N_3$. If one of the subsets, say A, equals L after finishing moving the leaves, then we select a leaf u that maximizes $\frac{p(u)}{c(u)}$, where $p(u) = |\{(\{x, y\}, u) \in T\}|$ and $c(u) = |\{(\{u, x\}, y) \in \mathcal{T}\}|$, and move u from A to either B or C. (This step is to ensure that none of the three subsets equals L.) It can be shown that this extra move does not reduce the value of $score(A, B, C)$. Since $score$ increases in every iteration of the **while**-loop, Step 2.1 is performed at most $12 \cdot |\mathcal{T}|$ times in total.

The next two lemmas are needed to analyze the approximation ratio of *Approximate*.

LEMMA 5.5. *When Algorithm LeafPartition terminates, we have* $5N_1 + 8N_2 + 12N_3 \geq 5 \cdot |\mathcal{T}|$.

Proof. (Sketch.) Write $score(A, B, C) = x \cdot N_1 + y \cdot N_2 + z \cdot N_3$. For each of the six possible ways of moving a leaf m from one of the subsets A, B, C to another, we derive a formula to express how *score* is affected. After *LeafPartition* is done, moving m will not increase the value of *score*, so, e.g., $score(A \setminus \{m\}, B \cup \{m\}, C) - score(A, B, C) \leq 0$. Then, by summing over all $m \in A$, we obtain an inequality I_{AB}. In the same way, we derive five inequalities I_{AC}, I_{BA}, I_{BC}, I_{CA}, and I_{CB}, and then add them to obtain $(z + 2y + x) \cdot N_0 + (2z - 6x) \cdot N_1 + (2z - 6y) \cdot N_2 + (2y + x - 5z) \cdot N_3 \leq 0$. By substituting $N_0 = |\mathcal{T}| - N_1 - N_2 - N_3$ and replacing $x = 4$, $y = 7$, $z = 12$, we get $5N_1 + 8N_2 + 12N_3 \geq 5 \cdot |\mathcal{T}|$. $\square$

```
Algorithm  LeafPartition
Input:   A set $\mathcal{T}$ of rooted triplets.
Output:  A partition of $L$ into three subsets $A, B, C$
         such that none of them equals $L$ and such
         that $5N_1 + 8N_2 + 12N_3 \geq 5 \cdot |\mathcal{T}|$.
1    Arbitrarily partition $L$ into three subsets $A, B, C$.
2    while moving a leaf $m$ from one subset to another
     increases $score(A, B, C) = 4N_1 + 7N_2 + 12N_3$ do
2.1          Move $m$ accordingly.
     endwhile
3    if one of the subsets $A, B, C$ equals $L$ then
3.1      Choose a leaf $u$ that maximizes $\frac{p(u)}{c(u)}$ and move $u$
         to another subset. Go to Step 2.
     endif
4    return $A, B, C$.
End  LeafPartition
```

```
Algorithm  Approximate
Input:   A set $\mathcal{T}$ of rooted triplets.
Output:  A galled network that is consistent with at
         least $\frac{5}{12} \cdot |\mathcal{T}|$ of the rooted triplets in $\mathcal{T}$.
1    Partition $L$ into $A, B, C$ using LeafPartition.
2    For $X \in \{A, B, C\}$, let $K_X = Approximate(\mathcal{T}|X)$.
3    For $X \in \{A, B, C\}$, generate a galled network
     $Network_X$ in which the root node is a split node
     whose hybrid node $h$ 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 the merge paths of $h$.
4    return the $Network_X$ among $X \in \{A, B, C\}$ that
     is consistent with the most rooted triplets in $\mathcal{T}$.
End  Approximate
```

Figure 6: An approximation algorithm for computing a galled network consistent with as many rooted triplets in $\mathcal{T}$ as possible.

Let $m(\mathcal{T})$ be the number of rooted triplets in $\mathcal{T}$ consistent with the network returned by $Approximate(\mathcal{T})$.

LEMMA 5.6. *If* $m(\mathcal{T}|Z) \geq q \cdot |\mathcal{T}|Z|$ *for every* $Z \in \{A, B, C\}$ *then* $m(\mathcal{T}) \geq q \cdot N_1 + \frac{2}{3} \cdot N_2 + N_3$.

Proof. Every rooted triplet in $Z_2(A, B, C)$ is consistent with two of $Network_A$, $Network_B$, and $Network_C$, and every rooted triplet in $Z_3(A, B, C)$ is consistent with all of these three networks. So, $Network_X$ returned by *Approximate* must be consistent with at least $\frac{2}{3} \cdot N_2 + N_3$ of the rooted triplets in $Z_2(A, B, C) \cup Z_3(A, B, C)$. Also, each of $Network_A$, $Network_B$, and $Network_C$ is consistent with $m(\mathcal{T}|A) + m(\mathcal{T}|B) + m(\mathcal{T}|C) \geq q \cdot (|\mathcal{T}|A| + |\mathcal{T}|B| + |\mathcal{T}|C|) = q \cdot N_1$ of the rooted triplets in $Z_1(A, B, C)$. Thus, in total, $Network_X$ is consistent with at least $q \cdot N_1 + \frac{2}{3} \cdot N_2 + N_3$ rooted triplets in $\mathcal{T}$. $\square$

THEOREM 5.2. $m(\mathcal{T}) \geq \frac{5}{12} \cdot |\mathcal{T}|$.

Proof. By induction on $|L|$. **Base case ($|L| = 3$):** Steps 3 and 4 of Algorithm *Approximate* construct a network consistent with at least 2/3 of the rooted triplets in $\mathcal{T}$, i.e., $m(\mathcal{T}) \geq \frac{5}{12} \cdot |\mathcal{T}|$. **Inductive case ($|L| > 3$):** Step 2 of *Approximate* recursively constructs three networks K_A, K_B, K_C for $\mathcal{T}|A$, $\mathcal{T}|B$, and $\mathcal{T}|C$, respectively. By the induction assumption, $m(\mathcal{T}|X) \geq \frac{5}{12} \cdot |\mathcal{T}|X|$ for each $X \in \{A, B, C\}$. By Lemmas 5.5 and 5.6, $m(\mathcal{T}) \geq \frac{5}{12} \cdot N_1 + \frac{2}{3} \cdot N_2 + N_3 \geq \frac{5}{12} \cdot |\mathcal{T}|$. □

Lastly, the algorithm's running time is given by:

Theorem 5.3. *The time complexity of Algorithm Approximate is $O(n \cdot |\mathcal{T}|^3)$.*

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 J. Comp.*, 10(3):405–421, 1981.

[2] 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.

[3] B. Chor, M. Hendy, and D. Penny. Analytic solutions for three-taxon ML$_{MC}$ trees with variable rates across sites. In *Proc. of the 1st Workshop on Algorithms in Bioinformatics* (WABI 2001), volume 2149 of *LNCS*, pages 204–213. Springer, 2001.

[4] C. Choy, J. Jansson, K. Sadakane, and W.-K. Sung. Computing the maximum agreement of phylogenetic networks. In *Proc. Computing: the 10th Australasian Theory Symposium* (CATS 2004), pages 33–45, 2004.

[5] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[6] L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. Inferring ordered trees from local constraints. In *Proc. Computing: the 4th Australasian Theory Symposium* (CATS'98), volume 20(3) of *Australian Computer Science Communications*, pages 67–76. Springer-Verlag Singapore, 1998.

[7] 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.

[8] D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proc. of Computational Systems Bioinformatics* (CSB2003), pages 363–374, 2003.

[9] 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.

[10] J. Holm, K. de Lichtenberg, and M. Thorup. Polylogarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, 2001.

[11] D. H. Huson, T. Dezulian, T. Klöpper, and M. Steel. Phylogenetic super-networks from partial trees. In *Proc. of the 4th Workshop on Algorithms in Bioinformatics* (WABI 2004), to appear.

[12] J. Jansson. On the complexity of inferring rooted evolutionary trees. In *Proc. of the Brazilian Symposium on Graphs, Algorithms, and Combinatorics* (GRACO 2001), volume 7 of *Electronic Notes in Discrete Mathematics*, pages 121–125. Elsevier, 2001.

[13] J. Jansson, J. H.-K. Ng, K. Sadakane, and W.-K. Sung. Rooted maximum agreement supertrees. In *Proc. Latin American Theoretical Informatics* (LATIN'04), volume 2976 of *LNCS*, pages 499–508. Springer, 2004.

[14] J. Jansson and W.-K. Sung. Inferring a level-1 phylogenetic network from a dense set of rooted triplets. In *Proc. of the 10th International Computing and Combinatorics Conference* (COCOON 2004), volume 3106 of *LNCS*, pages 462–471. Springer, 2004.

[15] T. Jiang, P. Kearney, and M. Li. A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM J. Comp.*, 30(6):1942–1961, 2001.

[16] S. Kannan, E. Lawler, and T. Warnow. Determining the evolutionary tree using experiments. *Journal of Algorithms*, 21(1):26–50, 1996.

[17] 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.

[18] L. Nakhleh, T. Warnow, and C. R. Linder. Reconstructing reticulate evolution in species – theory and practice. In *Proc. of the 8th Annual International Conference on Research in Computational Molecular Biology* (RECOMB 2004), pages 337–346, 2004.

[19] 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.

[20] M. P. Ng and N. C. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69(1–2):19–31, 1996.

[21] D. Posada and K. A. Crandall. Intraspecific gene genealogies: trees grafting into networks. *TRENDS in Ecology & Evolution*, 16(1):37–45, 2001.

[22] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.

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

[24] B. Y. Wu. Constructing the maximum consensus tree from rooted triples. *Journal of Combinatorial Optimization*, 8:29–39, 2004.