Inferring a Level-1 Phylogenetic Network from a Dense Set of Rooted Triplets

Jesper Jansson and Wing-Kin Sung

School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543
{jansson,ksung}@comp.nus.edu.sg

Abstract. Given a set $\mathcal{T}$ of rooted triplets with leaf set L, we consider the problem of determining whether there exists a phylogenetic network consistent with $\mathcal{T}$, and if so, constructing one. If no restrictions are placed on the hybrid nodes in the solution, the problem is trivially solved in polynomial time by a simple sorting network-based construction. For the more interesting (and biologically more motivated) case where the solution is required to be a level-1 phylogenetic network, we present an algorithm solving the problem in $O(n^6)$ time when $\mathcal{T}$ is dense (i.e., contains at least one rooted triplet for each cardinality three subset of L), where $n = |L|$. Note that the size of the input is $\Theta(n^3)$ if $\mathcal{T}$ is dense. We also give an $O(n^5)$-time algorithm for finding the set of *all* phylogenetic networks having a single hybrid node attached to exactly one leaf (and having no other hybrid nodes) that are consistent with a given dense set of rooted triplets.

1 Introduction

A phylogenetic network is a generalization of a phylogenetic tree in which internal nodes are allowed to have more than one parent. Phylogenetic networks are used to represent evolutionary relationships that cannot be adequately described in a single tree structure due to evolutionary events such as recombination, horizontal gene transfer, or hybrid speciation which suggest convergence between objects [8, 9, 17, 18, 20]. In fact, these types of events occur frequently in the evolutionary history of certain groups of organisms [17], but not much is known about the combinatorics related to phylogenetic networks [8]. Hence, to develop conceptual tools and efficient methods for computing with phylogenetic networks is an important issue.

Some methods for constructing and for comparing phylogenetic networks have been proposed recently [4, 8, 17, 18, 20]. In this paper, we consider the problem of constructing a phylogenetic network from a set of rooted triplets (see below for a formal problem definition). In particular, we assume that the input forms a *dense* set, meaning that the input contains at least one rooted triplet for each cardinality three subset of the objects being studied, and that the underlying phylogenetic network is a *level*-1 network, meaning that each biconnected component in the undirected version of the network contains at most one node

K.-Y. Chwa and J.I. Munro (Eds.): COCOON 2004, LNCS 3106, pp. 462–471, 2004.

which has two parents in the directed version of the network. The biological significance of level-1 phylogenetic networks, there referred to as *galled-trees*, is discussed in [8]. The rationale for assuming the input to consist of rooted triplets is that although computationally expensive methods for constructing reliable phylogenetic trees such as maximum likelihood are infeasible for large sets of objects, they can be applied to infer highly accurate trees for smaller, overlapping subsets of the objects (see, e.g., [3]). One may thus apply maximum likelihood to each cardinality three subset L' of the objects and then select the most likely rooted triplet for L' to get a dense input set[1]. Moreover, in some applications, the data obtained experimentally may already have the form of rooted triplets; for example, Sibley-Ahlquist-style DNA-DNA hybridization experiments (see [15]) can yield rooted triplets directly.

1.1 Definitions

A *rooted triplet* is a binary, rooted, unordered tree with three distinctly labeled leaves. The unique rooted triplet on 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 set $\mathcal{T}$ of rooted triplets is called *dense* if for each $\{x, y, z\} \subseteq L$, where L is the set of all leaves occurring in $\mathcal{T}$, at least one of the three possible rooted triplets $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ belongs to $\mathcal{T}$.

A *phylogenetic network* is a connected, rooted, simple, directed acyclic graph in which: (1) each node has outdegree at most 2; (2) each node has indegree 1 or 2, except the root node which has indegree 0; (3) no node has both indegree 1 and outdegree 1; and (4) all nodes with outdegree 0 are labeled by elements from a finite set L in such a way that no two nodes are assigned the same label. From here on, nodes of outdegree 0 are referred to as *leaves* and identified with their corresponding elements in L. We refer to nodes with indegree 2 as *hybrid nodes*. 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 a *level-f phylogenetic network* if every biconnected component in $\mathcal{U}(N)$ contains at most f nodes that are hybrid nodes in N. Note that if $f = 0$ then N is a tree.

We denote the set of leaves in a rooted triplet t or a phylogenetic network N by $\Lambda(t)$ or $\Lambda(N)$, respectively. A rooted triplet t is *consistent with* the phylogenetic network N if t is an induced subgraph of N. See Fig. 1 for an example. A set $\mathcal{T}$ of rooted triplets is *consistent with* N if every $t_i \in \mathcal{T}$ is consistent with N.

The problem we consider in this paper is: Given a set $\mathcal{T} = \{t_1, \ldots, t_k\}$ of rooted triplets, construct a level-1 phylogenetic network N with $\Lambda(N) =$

[1] A similar approach is used in the quartet method paradigm [14, 16] for reconstructing unrooted phylogenetic trees: first infer the unrooted topology of each cardinality four subset of the leaf set to obtain a complete set of quartets (unrooted, distinctly leaf-labeled trees each having four leaves and no nodes of degree two), then combine the quartets into an unrooted tree.

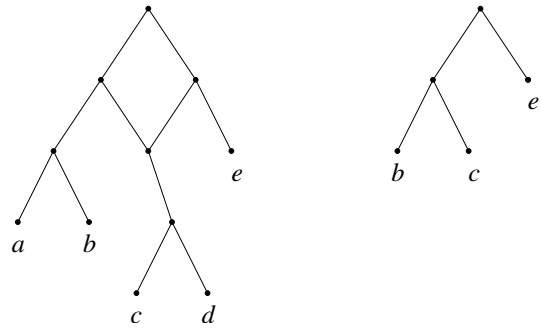

Fig. 1. Let N be the phylogenetic network on the left. The rooted triplet $(\{b, c\}, e)$ shown on the right is consistent with N. Note that $(\{c, e\}, b)$ is also consistent with N.

$\bigcup_{t_i \in \mathcal{T}} \Lambda(t_i)$ such that $\mathcal{T}$ is consistent with N, if such a network exists; otherwise, output *null*. Throughout this paper, L represents the leaf set $\bigcup_{t_i \in \mathcal{T}} \Lambda(t_i)$ in the problem definition above, and we write $n = |L|$ and $k = |\mathcal{T}|$. Note that $\binom{n}{3} \leq k \leq 3 \cdot \binom{n}{3}$, i.e., $k = \Theta(n^3)$ if the input is dense.

Finally, for any set $\mathcal{T}$ of rooted triplets and $L' \subseteq L$, we define $\mathcal{T} \mid L'$ as the subset of $\mathcal{T}$ consisting of all rooted triplets $(\{x, y\}, z)$ with $\{x, y, z\} \subseteq L'$.

1.2 Related Work

Aho, Sagiv, Szymanski, and Ullman [1] presented an $O(kn)$-time algorithm for determining whether a given set of k rooted triplets on n leaves is consistent with some rooted, distinctly leaf-labeled tree (i.e., a level-0 phylogenetic network), and if so, returning such a tree. Several years later, Henzinger, King, and Warnow [10] showed how to implement the algorithm of Aho *et al.* to run in $\min\{O(kn^{0.5}), O(k + n^2 \log n)\}$ time. Gąsieniec, Jansson, Lingas, and Östlin [6] considered a version of the problem where the leaves in the output tree are required to comply with a left-to-right leaf ordering given as part of the input. Related optimization problems where the objective is to construct a rooted tree consistent with the maximum number of rooted triplets in the input or to find a maximum cardinality subset L' of L such that $\mathcal{T} \mid L'$ is consistent with some tree have been studied in [2, 6, 7, 12] and [13], respectively.

The analog of the problem considered by Aho *et al.* for *unrooted* trees is NP-hard, even if all of the input trees are quartets [19]. Fortunately, certain useful optimization problems involving quartets can be approximated efficiently [14, 16]. For a survey on quartet-based methods for inferring unrooted phylogenetic trees and related computational complexity results, see [16].

Nakhleh, Warnow, and Linder [17] gave an algorithm for reconstructing a level-1 phylogenetic network from two distinctly leaf-labeled, binary, rooted, unordered trees with identical leaf sets. It runs in time which is polynomial in the number of leaves and the number of hybrid nodes in the underlying phylogenetic

network. They also considered the case where the two input trees may contain errors but where only one hybrid node is allowed.

We remark that the deterministic algorithm for dynamic graph connectivity employed in the algorithm of Henzinger *et al.* [10] mentioned above can in fact be replaced with a more recent one due to Holm, de Lichtenberg, and Thorup [11] to yield the following improvement.

Lemma 1. *[13] The algorithm of Aho* et al. *can be implemented to run in* $\min\{O(k \log^2 n),\ O(k + n^2 \log n)\}$ *time.*

1.3 Our Results and Organization of the Paper

We observe that if no restriction is placed on the level of the phylogenetic network, then the problem can be trivially solved using a sorting network-based construction in Section 2. Next, in Section 3, we present an $O(n^5)$-time algorithm called *OneHybridLeaf* for inferring all phylogenetic networks with one hybrid node to which exactly one leaf is attached that are consistent with a given dense set of rooted triplets. This algorithm is subsequently used in Section 4, where we give a more general algorithm called *LevelOne* for constructing a level-1 phylogenetic network consistent with $\mathcal{T}$ (if one exists) in $O(n^6)$ time when $\mathcal{T}$ is dense.

2 Constructing an Unrestricted Phylogenetic Network

Given a set $\mathcal{T}$ of rooted triplets, we can always construct a level-f phylogenetic network N where f is unrestricted such that N is consistent with $\mathcal{T}$. Moreover, the construction can be carried out in time which is polynomial in the size of $\mathcal{T}$ as follows. Let P be any sorting network (see, e.g., [5]) for n elements with a polynomial number p of comparator stages. Build a directed acyclic graph Q from P with $(p+1) \cdot n$ nodes $\{Q_{i,j} \mid 1 \leq i \leq p+1,\ 1 \leq j \leq n\}$ such that there is a directed edge $(Q_{i,j},\ Q_{i+1,j})$ for every $1 \leq i \leq p$ and $1 \leq j \leq n$, and two directed edges $(Q_{i,j},\ Q_{i+1,k})$ and $(Q_{i,k},\ Q_{i+1,j})$ for every comparator (j, k) at stage i in P for $1 \leq i \leq p$. Then, for $1 \leq j \leq n-1$, add the directed edge $(Q_{1,j},\ Q_{1,j+1})$. See Fig. 2. Finally, distinctly label the nodes $\{Q_{p+1,j} \mid 1 \leq j \leq n\}$ by L, and for each node in Q having indegree 1 and outdegree 1 (if any), contract its outgoing edge to obtain N. It is easy to show that for any $\{x, y, z\} \subseteq L$, all three of $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ are consistent with N.

3 Constructing All Phylogenetic Networks Having One Hybrid Node with One Attached Leaf

This section presents an algorithm called *OneHybridLeaf* for inferring the set of all phylogenetic networks having a single hybrid node attached to exactly one leaf (and having no other hybrid nodes) which are consistent with a given set $\mathcal{T}$ of rooted triplets. This algorithm is later used as a subroutine by the main algorithm in Section 4. *OneHybridLeaf* assumes that its given set $\mathcal{T}$ of rooted triplets is dense. We first note the following.

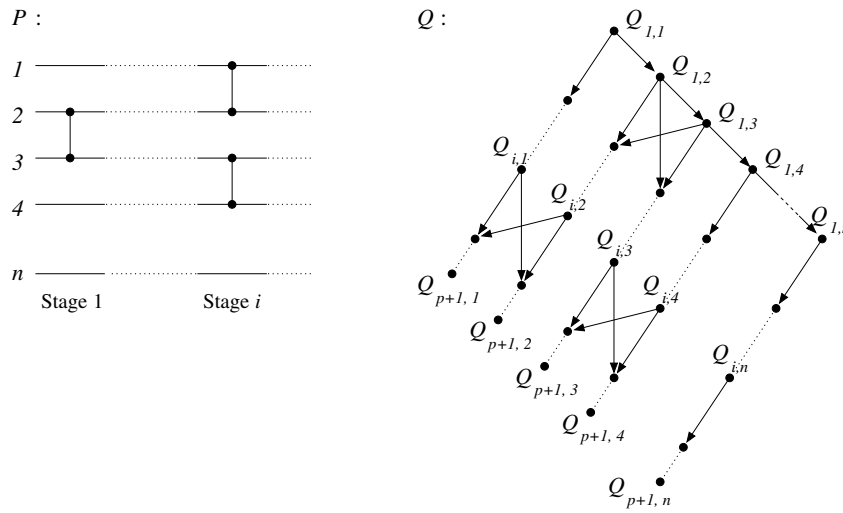

Fig. 2. The sorting network P on the left yields a directed acyclic graph Q.

Lemma 2. *Let N be any phylogenetic network consistent with a set $\mathcal{T}$ of rooted triplets with leaf set L such that N has a hybrid node h to which exactly one leaf c is attached and N has no other hybrid nodes. If h and c and all their incident edges are deleted and then, for every node with outdegree* 1 *and indegree less than* 2*, its outgoing edge is contracted, then the resulting graph is a binary tree consistent with $\mathcal{T} \mid (L \setminus \{c\})$.*

Lemma 3. *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 R exists then it must be binary.*

Proof. Suppose there exist two unordered, distinctly leaf-labeled trees R and R' consistent with $\mathcal{T}$ such that $R \neq R'$. Then, for some $x, y, z \in L$, $(\{x, y\}, z)$ is consistent with R while $(\{x, z\}, y)$ is consistent with R'. Since $\mathcal{T}'$ is dense, at least one of $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ belongs to $\mathcal{T}$. This yields a contradiction in all cases because R cannot be consistent with $(\{x, z\}, y)$ or $(\{y, z\}, x)$ and R' cannot be consistent with $(\{x, y\}, z)$ or $(\{y, z\}, x)$ since R and R' are trees.

Next, suppose R is not binary. Then R has a node u with degree greater than two. Let x, y, and z be leaves from three different subtrees rooted at children of u. $\mathcal{T}$ is dense, so at least one of $(\{x, y\}, z)$, $(\{x, z\}, y)$, and $(\{y, z\}, x)$ belongs to $\mathcal{T}$. But none of these three rooted triplets is consistent with R. Contradiction. □

Our algorithm *OneHybridLeaf* is shown in Fig. 3. It tests every $c \in L$ as the leaf attached to the hybrid node. For each such candidate c, it first calls a procedure *BuildTree* to obtain a binary tree R which is consistent with all rooted triplets in $\mathcal{T}$ that do not involve the leaf c, if such a tree exists. ($\mathcal{T}$ is dense, so the set $\mathcal{T} \mid (L \setminus \{c\})$ is also dense. Thus, Lemma 3 ensures that if R exists then it

Algorithm *OneHybridLeaf*
Input: A dense set $\mathcal{T}$ of rooted triplets with leaf set L.
Output: The set of all phylogenetic networks which are consistent with $\mathcal{T}$ having exactly one hybrid node to which there is exactly one leaf attached.

1 Set $\mathcal{N} = \emptyset$.
2 **for** each $c \in L$ **do**
2.1 Let $R = BuildTree(\mathcal{T} \mid (L \setminus \{c\}))$.
2.2 **if** $R \neq null$ **then**
2.3 **for** every pair of nodes u and v in R, where $u \neq v$ and v is not the root of R **do**
2.3.1 Form a phylogenetic network N from R as follows. Create three new internal nodes p, q, and h, insert the four edges (p, u), (p, h), (q, v), and (q, h), and attach a leaf child labeled by c to h. Replace the edge (v_0, v) leading to v by the edge (v_0, q). If u is not the root then replace u's incoming edge (u_0, u) by the edge (u_0, p).
2.3.2 If N is consistent with all rooted triplets in $\mathcal{T}$ involving the leaf c then let $\mathcal{N} = \mathcal{N} \cup \{N\}$.
endfor
endfor
3 **return** $\mathcal{N}$.
End *OneHybridLeaf*

Fig. 3. Constructing phylogenetic networks with one hybrid node.

is uniquely determined and binary.) Then, it tries all possible ways to obtain a phylogenetic network from R by inserting a hybrid node h attached to the leaf c, and keeps all resulting networks which are also consistent with the rest of $\mathcal{T}$. By Lemma 2, all valid networks will be found by *OneHybridLeaf*.

To implement *BuildTree*, we use the fast version of the algorithm of Aho *et al.* referred to in Lemma 1. If L contains at least four elements then $BuildTree(\mathcal{T} \mid (L \setminus \{c\}))$ is the algorithm of Aho *et al.* applied to $\mathcal{T} \mid (L \setminus \{c\})$ (we may assume it returns *null* if it fails). For the case $|L| = 3$, the set $\mathcal{T} \mid (L \setminus \{c\})$ is empty and we simply let $BuildTree(\mathcal{T} \mid (L \setminus \{c\}))$ return a tree with the two leaves in $L \setminus \{c\}$.

Lemma 4. *The time complexity of Algorithm OneHybridLeaf is* $O(n^5)$.

Proof. Step 2 iterates Steps 2.1–2.3 n times. In each iteration, Step 2.1 takes $O(k + n^2 \log n)$ time by Lemma 1. The inner **for**-loop (Step 2.3) considers $O(n^2)$ pairs of nodes of R; for each such node pair, Step 2.3.1 takes $O(1)$ time and Step 2.3.2 takes $O(n^2)$ time. In total, Step 2.3 uses $O(n^2 \cdot (1 + n^2)) = O(n^4)$ time, so Step 2 takes $O(n \cdot (k + n^2 \log n + n^4))$ time. Furthermore, $k = |\mathcal{T}| = O(n^3)$. Thus, the total running time of Algorithm *OneHybridLeaf* is $O(n^5)$. □

4 Constructing a Level-1 Phylogenetic Network

Here, we present an algorithm called *LevelOne* for inferring a level-1 phylogenetic network (if one exists) consistent with a given dense set $\mathcal{T}$ of rooted triplets. The

basic idea of our algorithm is to partition the leaf set of $\mathcal{T}$ into disjoint subsets which we call SN-sets, run *LevelOne* recursively to construct a level-1 network for each SN-set, and then apply Algorithm *OneHybridLeaf* from Section 3 to combine the computed networks for the SN-sets into one level-1 network.

We first introduce the concept of an SN-set. Let L be the leaf set of $\mathcal{T}$. For any $X \subseteq L$, define the set $SN(X)$ recursively as $SN(X \cup \{c\})$ if there exists some $c \in L \setminus X$ and $x_1, x_2 \in X$ such that $(\{x_1, c\}, x_2) \in \mathcal{T}$, and as X otherwise. Below, we study some properties of the SN-sets.

Lemma 5. *$SN(\{x, y\})$ for any $x, y \in L$ is computable in $O(n^3)$ time.*

Proof. If $x = y$ then $SN(\{x, y\}) = \{x\}$ can be obtained in $O(1)$ time. If $x \neq y$ then $SN(\{x, y\})$ can be computed by calling Algorithm *ComputeSN*(x, y) shown in Fig. 4. Initially, the algorithm sets $X = \{x\}$ and $Z = \{y\}$. Then, while Z is nonempty, it selects any $z \in Z$, augments Z with all leaves c not already in $X \cup Z$ such that $(\{a, c\}, z)$ or $(\{z, c\}, a) \in \mathcal{T}$ for some $a \in X$, and finally removes z from Z and inserts z into X. To analyze the time complexity of Algorithm *ComputeSN*, observe that one leaf is transferred from Z to X in each iteration of the **while**-loop and that a leaf which has been moved to X can never be moved back to Z, so Steps 2.1–2.3 are iterated at most $n - 1$ times. Inside the **while**-loop, the algorithm scans $O(n^2)$ rooted triplets at most once to augment Z. The total running time of *ComputeSN* is therefore $O(n^3)$. □

Algorithm *ComputeSN*
Input: A dense set $\mathcal{T}$ of rooted triplets with leaf set L. Two leaves $x, y \in L$.
Output: $SN(\{x, y\})$.

1 Set $X = \{x\}$ and $Z = \{y\}$.
2 **while** $Z \neq \emptyset$ **do**
2.1 Let z be any element in Z.
2.2 **for** every $a \in X$ **do**
If there exists some $c \in L \setminus (X \cup Z)$ such that $(\{a, c\}, z) \in \mathcal{T}$ or $(\{z, c\}, a) \in \mathcal{T}$ then $Z = Z \cup \{c\}$.
endfor
2.3 Set $X = X \cup \{z\}$ and $Z = Z \setminus \{z\}$.
endwhile
3 **return** X.
End *ComputeSN*

Fig. 4. Computing $SN(\{x, y\})$.

Lemma 6. *If $\mathcal{T}$ is dense then for any $A, B \subseteq L$, $SN(A) \cap SN(B)$ equals $\emptyset$, $SN(A)$, or $SN(B)$.*

Proof. Suppose on the contrary that $z_1, z_2 \in SN(A)$, $z_2, z_3 \in SN(B)$, $z_3 \notin SN(A)$, and $z_1 \notin SN(B)$. Consider the rooted triplet on the three leaves z_1, z_2, and z_3. Since $\mathcal{T}$ is dense, at least one of the following three cases must occur:

- Case 1: $(\{z_2, z_3\}, z_1) \in \mathcal{T}$. Then, by definition, $z_3 \in SN(A)$.
- Case 2: $(\{z_1, z_3\}, z_2) \in \mathcal{T}$. Then, by definition, $z_3 \in SN(A)$.
- Case 3: $(\{z_1, z_2\}, z_3) \in \mathcal{T}$. Then, by definition, $z_1 \in SN(B)$.

In each of the three cases, we have a contradiction. Thus, the lemma follows. □

In particular, Lemma 6 holds for all subsets of L of cardinality one or two.

For any $x_1, x_2 \in L$ (possibly with $x_1 = x_2$), $SN(\{x_1, x_2\})$ is called *trivial* if $SN(\{x_1, x_2\}) = L$, and $SN(\{x_1, x_2\})$ is called *maximal* if it is nontrivial and not a proper subset of any nontrivial $SN(\{y_1, y_2\})$, where $y_1, y_2 \in L$. Let $\mathcal{SN}$ be the set of all maximal SN-sets of the form $SN(\{x_1, x_2\})$, where possibly $x_1 = x_2$. Since $\mathcal{T}$ is dense, $\mathcal{SN}$ forms a partition of the set L by Lemma 6. Furthermore, $\mathcal{SN}$ is uniquely determined. Write $\mathcal{SN} = \{SN_1, SN_2, \ldots, SN_q\}$ and introduce q new symbols $\alpha_1, \alpha_2, \ldots, \alpha_q$. (Observe that $q \geq 2$ if $|L| \geq 2$.) We define a function f as follows. For every $x \in L$, let $f(x) = \alpha_i$ if $x \in SN_i$. Let $\mathcal{T}'$ be the set $\big\{(\{f(x), f(y)\}, f(z)) : (\{x, y\}, z) \in \mathcal{T}$ and $f(x), f(y), f(z)$ all differ$\big\}$.

Lemma 7. *Suppose $\mathcal{T}$ is consistent with a level-1 phylogenetic network. If $q = 2$ then the tree distinctly leaf-labeled by α_1 and α_2 is consistent with $\mathcal{T}'$. If $q \geq 3$ then there exists a phylogenetic network having a single hybrid node attached to exactly one leaf (and having no other hybrid nodes) that is consistent with $\mathcal{T}'$.*

We also have:

Lemma 8. *Suppose $\mathcal{T}'$ is consistent with a level-1 phylogenetic network N' with leaf set $\{\alpha_1, \ldots, \alpha_q\}$. Let N be a level-1 network obtained from N' by replacing each α_i by a level-1 network N_i with leaf set SN_i consistent with $\mathcal{T} \mid SN_i$. Then N is consistent with $\mathcal{T}$.*

Proof. Let t be any rooted triplet in $\mathcal{T}$ and write $t = (\{x, y\}, z)$. If $x \in SN_i$, $y \in SN_j$, and $z \in SN_k$, where i, j, k all differ, then t is consistent with N (otherwise, $t' = (\{f(x), f(y)\}, f(z)) = (\{\alpha_i, \alpha_j\}, \alpha_k)$ cannot be consistent with N' which is a contradiction since $t' \in \mathcal{T}'$). If $x, y \in SN_i$ and $z \in SN_j$ with $i \neq j$ then t is consistent with N by the construction of N. The case $x, z \in SN_i$ and $y \in SN_j$ (or symmetrically, $y, z \in SN_i$ and $x \in SN_j$) with $i \neq j$ is not possible since $x, z \in SN_i$ implies $y \in SN_i$. If x, y, z belong to the same SN_i then t is consistent with N_i and therefore with N. In all cases, t is consistent with N. □

Our main algorithm *LevelOne* is listed in Fig. 5. Its correctness follows from Lemmas 7 and 8.

Theorem 1. *When $\mathcal{T}$ is dense, we can determine if there exists a level-1 phylogenetic network consistent with $\mathcal{T}$, and if so construct one, in $O(n^6)$ time.*

Proof. Apply Algorithm *LevelOne* to $\mathcal{T}$. For any $L' \subseteq L$, let $g(L')$ be the running time of *LevelOne*$(\mathcal{T} \mid L')$. In Step 1 of the algorithm, we compute $SN(\{x_1, x_2\})$ for the n^2 pairs (x_1, x_2) in $L \times L$. By Lemma 5, Step 1 takes $O(n^5)$ time. Step 2 can be performed in $O(n^3)$ time, and Step 3 takes $\sum_{SN_i \in \mathcal{SN}} g(SN_i)$ time. Step 5 can be done in $O(n^5)$ time according to Lemma 4. In total, we have $g(L) = \sum_{SN_i \in \mathcal{SN}} g(SN_i) + O(n^5)$. Since all sets in $\mathcal{SN}$ are disjoint, $g(L) = O(n^6)$. □

Algorithm *LevelOne*
Input: A dense set $\mathcal{T}$ of rooted triplets with leaf set L.
Output: A level-1 network N consistent with $\mathcal{T}$, if one exists; otherwise, *null*.
1 **for** every $x_1 \in L$ and $x_2 \in L$ (including $x_1 = x_2$) **do**
Compute $SN(\{x_1, x_2\})$.
endfor
2 Let $\mathcal{SN} = \{SN_1, SN_2, \ldots, SN_q\}$ be the set of all maximal $SN(\{x_1, x_2\})$.
3 **for** every $SN_i \in \mathcal{SN}$ **do**
If $|SN_i| \geq 3$ then $N_i = LevelOne(\mathcal{T} \mid SN_i)$; else, let N_i be a tree distinctly leaf-labeled by SN_i.
endfor
4 If N_i for any $i \in \{1, \ldots, q\}$ equals *null* then **return** *null*.
5 If $q = 2$ then let N be a network with a root node connected to N_1 and N_2. Otherwise ($q \geq 3$), build $\mathcal{T}'$ from $\mathcal{T}$, compute $\mathcal{N} = OneHybridLeaf(\mathcal{T}')$, and check if $\mathcal{N}$ is empty; if yes then let $N = null$, else select any $N' \in \mathcal{N}$ and form a network N by replacing each α_i in N' with N_i.
6 **return** N.
End *LevelOne*

Fig. 5. Constructing a level-1 phylogenetic network.

Algorithm *LevelOne* can be modified to return *all* level-1 phylogenetic networks consistent with $\mathcal{T}$ by utilizing all the possible topologies returned by *OneHybridLeaf*. However, the running time may then become exponential since some inputs are consistent with an exponential number of different level-1 networks. (At each recursion level, although the partition of the leaves into $\mathcal{SN}$ is unique when the input is dense, there may be more than one way to merge the recursively computed subnetworks for the SN-sets into a valid network.)

5 Conclusion

This paper presents a polynomial-time algorithm for inferring a level-1 phylogenetic network from a dense set of rooted triplets. This problem is not only interesting from a combinatorial point of view, but also biologically sound since rooted triplets can be obtained accurately by using maximum likelihood or directly through experiments. In the future, we plan to further improve the time complexity of our main algorithm and to investigate the computational complexity of the problem when $\mathcal{T}$ is not dense. Also, we would like to know if it is possible to construct a level-f phylogenetic network from a dense set of rooted triplets in polynomial time for any constant $f > 1$.

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. 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. of Computing: the 10th Australasian Theory Symposium* (CATS 2004), pages 33–45. Elsevier, 2004.
5. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, Massachusetts, 1990.
6. L. Gąsieniec, J. Jansson, A. Lingas, and A. Östlin. Inferring ordered trees from local constraints. In *Proc. of 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*, 3:183–197, 1999.
8. D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proc. of the Computational Systems Bioinformatics Conference* (CSB2003), pages 363–374, 2003.
9. J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98(2):185–200, 1990.
10. 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.
11. J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, 2001.
12. J. Jansson. On the complexity of inferring rooted evolutionary trees. In *Proc. of the Brazilian Symp. on Graphs, Algorithms, and Combinatorics* (GRACO'01), 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. of Latin American Theoretical Informatics* (LATIN 2004), volume 2976 of *LNCS*, pages 499–508, 2004.
14. T. Jiang, P. Kearney, and M. Li. A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM Journal on Computing*, 30(6):1942–1961, 2001.
15. S. Kannan, E. Lawler, and T. Warnow. Determining the evolutionary tree using experiments. *Journal of Algorithms*, 21(1):26–50, 1996.
16. 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.
17. 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), to appear.
18. D. Posada and K. A. Crandall. Intraspecific gene genealogies: trees grafting into networks. *TRENDS in Ecology & Evolution*, 16(1):37–45, 2001.
19. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992.
20. L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.