A First Step Toward Computing All Hybridization Networks For Two Rooted Binary Phylogenetic Trees

Celine Scornavacca, Simone Linz, Benjamin Albrecht

A first step towards computing all hybridization networks for two rooted binary phylogenetic trees

Celine Scornavacca[*,†] Simone Linz[*,‡]

Benjamin Albrecht [§]

February 10, 2015

Abstract

Recently, considerable effort has been put into developing fast algorithms to reconstruct a rooted phylogenetic network that explains two rooted phylogenetic trees and has a minimum number of hybridization vertices. With the standard approach to tackle this problem being combinatorial, the reconstructed network is rarely unique. From

[*]Equally contributing authors.

[†]C. Scornavacca is at the ISEM, UMR 5554, Université Montpellier II. Montpellier, France. celine.scornavacca@univ-mont2.fr

[‡]Simone Linz is at the Biomathematics Research Centre, University of Canterbury. Christchurch, New Zealand linz@Informatik.uni-tuebingen.de

[§]Benjamin Albrecht is at the Institut of Informatics, Ludwig Maximilians University. Munich, Germany. benjamin.albrecht@bio.ifi.lmu.de

a biological point of view, it is therefore of importance to not only compute one network, but *all* possible networks. In this paper, we make a first step towards approaching this goal by presenting the first algorithm—called allMAAFs—that calculates all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees on the same set of taxa.

Keywords: Directed acyclic graphsHybridizationMaximum-acyclic-agreement forests-Bounded searchPhylogenetics

1 Introduction

Over the last decade, significant progress in phylogenetic studies has been achieved by combining the expertise acquired in the fields of biology, computer science, and mathematics. As for the latter, combinatorics is becoming increasingly important in approaching many problems in the context of reticulate evolution (e.g., see [**?** **?**] for two excellent reviews) which is an umbrella term for processes such as horizontal gene transfer, hybridization, and recombination. To analyze reticulation in evolution, the graph-theoretic concept of an agreement forest for two rooted phylogenetic trees has attracted much attention (e.g. [**?** **?** **?** **?** **?**]). However, most approaches that make use of this concept aim at quantifying the amount of reticulation that is needed to simultaneously explain a set of rooted phylogenetic trees. Thus, one is primarily interested in the number of horizontal gene transfer, hybridization, or recombination events that occurred during the evolution of a set of present-day species. Consequently, these approaches do not explicitly

construct a rooted phylogenetic network that explains a set of phylogenetic trees. Nevertheless, this is desirable from a biological point of view because such a network intuitively indicates how species may have evolved by means of speciation and reticulation. While each vertex of a phylogenetic tree has exactly one direct ancestor, a vertex of a phylogenetic networks may have more than one such ancestor; thereby indicating that the genome of the underlying species is a combination of the genomes of distinct parental species. Generically, we refer to such a vertex as a reticulation vertex or, more specific in the context of hybridization, as a hybridization vertex. Since reticulation events are assumed to be significantly less frequent than speciation events, current research aims at constructing a rooted phylogenetic network that explains a set of rooted phylogenetic trees and whose number of reticulation vertices is minimized.

For the purpose of the introduction, think of a so-called *maximum-acyclic-agreement forest* $\mathcal{F}$ for two rooted binary phylogenetic trees S and T as a small collection of vertex-disjoint rooted subtrees that are common to S and T (for details, see Section 2). It is well-known that the size of $\mathcal{F}$ minus 1 equates to the minimum number of hybridization events that are needed to explain S and T [**?**]. Furthermore, there exists an algorithm—called HybridPhylogeny [**?**]—that glues together the elements of $\mathcal{F}$ by introducing new edges such that the resulting graph is a rooted phylogenetic network that explains S and T and has $|\mathcal{F}| - 1$ hybridization vertices. However, until now, HybridPhylogeny, has not found its way into many practical applications that are concerned with reconstructing the evolutionary history for a set of species whose past is likely to include hybridization. This might

be due to the fact that the reconstructed phylogenetic network is rarely unique because the gluing step can often be done in a number of different ways. Furthermore, given two rooted binary phylogenetic trees S and T, a maximum-acyclic-agreement forest for S and T is rarely unique. Given these hurdles, an appealing open problem is the reconstruction of *all* rooted phylogenetic networks that explain a pair of rooted phylogenetic trees and whose number of hybridization vertices is minimized. Once having calculated the entire solution space of these networks, one can then for example apply statistical methods or additional biological knowledge to decide which of the phylogenetic network in this space is most likely to be the correct one.

In this paper, we focus on a first step to reach this goal. In particular, we give the first non-naive algorithm—called allMAAFs—that is based on a bounded-search type idea and calculates all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees S and T on the same set of taxa. With the underlying optimization problem being NP-hard [**?**] and fixed-parameter tractable [**?**], the running time of allMAAFs is exponential. More precisely, we will see in Section 5 that the running time of allMAAFs, which is $O(3^n)$, can be improved to $O(3^{14k}+p(n))$ by applying the kernelization rules of [**?**], where n is the number of leaves in S and T, $p(n)$ is some polynomial function that depends on n, and k is the minimum number of hybridization events needed to explain S and T.

While the description of allMAAFs is slightly complex in comparison to other straight-forward approaches (for more details, see Section 5), it has been shown in a recent paper by **?**], which contains the description of a practical implementation of allMAAFs but without providing any

theoretical background or mathematical justifications, that this algorithm is remarkably quick in practice. Therefore, this paper also aims at establishing the correctness of the algorithm presented in [**?**].

The paper is organized as follows. The next section contains preliminaries and some well-known results from the phylogenetics literature. Section 3 describes the algorithm allMAAFs that calculates all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees. Its pseudocode is also given in this section. Subsequently, in Section 4, we establish the correctness of allMAAFs and give its running time in Section 5. We finish the paper with some concluding remarks in Section 6.

2 Preliminaries

In this section, we give some preliminary definitions that are used throughout this paper. Notation and terminology on phylogenetic trees and networks follow [**?**] and [**?**], respectively.

Phylogenetic trees. A *rooted phylogenetic $\mathcal{X}$-tree* T is a connected graph with no (undirected) cycle, no vertices of degree 2, except for the root which has degree at least 2, and such that each element of $\mathcal{X}$ labels a leaf of T. The set $\mathcal{X}$ represents a collection of present-day taxa and internal vertices represent putative speciation events. A rooted phylogenetic $\mathcal{X}$-tree T is said to be *binary* if its root vertex has degree two while all other interior vertices have degree three. We denote the edge set of T by $E(T)$. The taxa set $\mathcal{X}$ of T is called the *label set* of T and is frequently denoted by $\mathcal{L}(T)$.

Furthermore, let v be a vertex of T. We denote by $\mathcal{L}(v)$ the label set of the rooted phylogenetic tree with root v that has been obtained from T by deleting the edge ending in v. Lastly, let $\mathcal{F}$ be a set of rooted phylogenetic trees. Similarly to $\mathcal{L}(T)$, we use $\mathcal{L}(\mathcal{F})$ to denote the union of leaf labels over all elements in $\mathcal{F}$.

We next introduce several types of subtrees that will play an important role in this paper. Let T be a rooted phylogenetic $\mathcal{X}$-tree, and let $\mathcal{X}' \subset \mathcal{X}$ be a subset of $\mathcal{X}$. We use $T(\mathcal{X}')$ to denote the minimal connected subgraph of T that contains all leaves that are labeled by elements of $\mathcal{X}'$. Furthermore, the *restriction of* T *to* $\mathcal{X}'$, denoted by $T|_{\mathcal{X}'}$, is defined as the rooted phylogenetic tree that has been obtained from $T(\mathcal{X}')$ by suppressing all non-root degree-2 vertices. Lastly, we say that a subtree of T is *pendant* if it can be detached from T by deleting a single edge.

Now, let T be a rooted binary phylogenetic $\mathcal{X}$-tree, and let $\mathcal{X}'$ be a subset of $\mathcal{X}$. Then, the *lowest common ancestor* of $\mathcal{X}'$ in T is the vertex v in T with $\mathcal{X}' \subseteq \mathcal{L}(v)$ such that there exists no vertex v' in T with $\mathcal{X}' \subseteq \mathcal{L}(v')$ and $\mathcal{L}(v') \subset \mathcal{L}(v)$. We denote v by $\mathrm{lca}_T(\mathcal{X}')$.

Hybridization networks. Let $\mathcal{X}$ be a finite set of taxa. A *rooted phylogenetic network* on $\mathcal{X}$ is a rooted acyclic digraph with no vertex of both indegree and outdegree one and whose leaves are bijectively labeled by elements of $\mathcal{X}$. Since this paper is concerned with hybridization as a representative of reticulation, we will often refer to a phylogenetic network as a *hybridization network*. Each internal vertex of a hybridization network

with indegree 1 represents a putative speciation event while each vertex with indegree of at least 2 represents a hybridization event and, therefore, a species whose genome is a chimaera of its parents' genomes. Generically, we call a vertex of the latter type a *hybridization vertex* and each edge that enters a hybridization vertex a *hybridization edge*.

To quantify the number of hybridization events, the *hybridization number* of N, denoted by $h(N)$, is defined as

$$h(N) = \sum_{v \in V(N): \delta^-(v) > 0} (\delta^-(v) - 1) = |E(N)| - |V(N)| + 1,$$

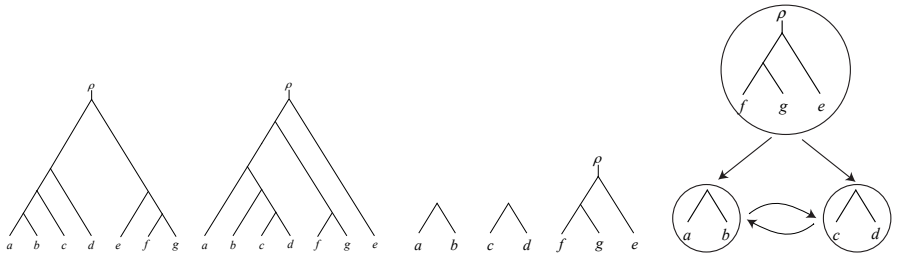
where $V(N)$ and $E(N)$ denote respectively the vertex and edge set of N and $\delta^-(v)$ the indegree of v. Note that, if N is a rooted phylogenetic tree, then $h(N) = 0$, and if $\delta^-(v)$ is at most 2 for each vertex $v \in V(N)$, then $h(N)$ is equal to the total number of hybridization vertices of N.

Now, let N be a phylogenetic network on $\mathcal{X}$, and let T be a rooted binary phylogenetic $\mathcal{X}'$-tree with $\mathcal{X}' \subseteq \mathcal{X}$. We say that T is *displayed* by N if T can be obtained from N by deleting a subset of its edges and any resulting degree-0 vertices, and then contracting edges. Intuitively, if N displays T, then all of the ancestral relationships visualized by T are visualized by N. In the remainder of this paper, we will consider the case where $\mathcal{T}$ is composed of two rooted binary phylogenetic trees.

Extending the definition of the hybridization number to two rooted binary phylogenetic $\mathcal{X}$-trees S and T, we set

$$h(S,T) = \min\{h(N) : N \text{ is a hybridization network that displays } S \text{ and } T\}.$$

Calculating $h(S,T)$ for two rooted binary phylogenetic $\mathcal{X}$-trees has been



(i) (ii) (iii)

Figure 1: (i) Two phylogenetic trees S and T on $\mathcal{X} = \{a, b, c, d, e, f, g\}$. (ii) An agreement forest $\mathcal{F}$ for S and T. (iii) The graph $AG(S, T, \mathcal{F})$. Since this graph contains a directed cycle, $\mathcal{F}$ is *not* an acyclic-agreement forest for S and T.

shown to be NP-hard [**?**].

Forests. Let T be a rooted binary phylogenetic $\mathcal{X}$-tree whose edge set is $E(T)$. For the purpose of the upcoming definitions and, indeed, much of the paper, we regard the root of T as a vertex labeled ρ at the end of a pendant edge adjoined to the original root of T. For an example of two such trees, see Figure 1(i). Furthermore, we view ρ as an element of the label set of T; thus $\mathcal{L}(T) = \mathcal{X} \cup \{\rho\}$. Any collection of rooted binary phylogenetic trees whose union of label sets is $\mathcal{L}(T)$ is a *forest on* $\mathcal{L}(T)$. Furthermore, we say that a set $\mathcal{F} = \{F_0, F_1, \ldots, F_k\}$ of rooted binary phylogenetic trees, with $|\mathcal{F}|$ referred to as the *size* of $\mathcal{F}$, is a *forest for* T if $\mathcal{F}$ can be obtained from T by deleting a k-sized subset E of $E(T)$ and, subsequently, suppressing vertices with both indegree and outdegree 1. To ease reading, we write $\mathcal{F} = T - E$ if $\mathcal{F}$ can be obtained in this way. Obviously, in the same way, we obtain a new forest $\mathcal{F}' = \{F'_0, F'_1, \ldots, F'_{k'}\}$ for T from $\mathcal{F}$ by deleting a k'-sized subset E' of

$\bigcup_{F_i \in \mathcal{F}} E(F_i)$ and, again, suppressing vertices with both indegree and outdegree 1. Similarly to the above, we write $\mathcal{F}' = \mathcal{F} - E'$ if $\mathcal{F}'$ can be obtained in this way. Now, let $\mathcal{F}$ be a forest for a rooted binary phylogenetic $\mathcal{X}$-tree. We use $\overline{\mathcal{F}}$ to denote the forest obtained from $\mathcal{F}$ by deleting all of its isolated vertices and, additionally, the element that contains the vertex labeled ρ if it contains at most one edge. Lastly, for two leaf vertices a and c with labels $\mathcal{L}(a)$ and $\mathcal{L}(c)$ respectively, we write $a \sim_{\mathcal{F}} c$ if there exists an element in $\mathcal{F}$ that contains a leaf labeled with $\mathcal{L}(a)$ and a distinct leaf labeled with $\mathcal{L}(c)$, otherwise, we write $a \nsim_{\mathcal{F}} c$.

Let S and T be two rooted binary phylogenetic $\mathcal{X}$-trees. A set $\mathcal{F} = \{F_\rho, F_1, F_2, \ldots, F_k\}$ of rooted phylogenetic trees is an *agreement forest* for S and T if $\mathcal{F}$ is a forest for S and T, and $\rho \in \mathcal{L}(F_\rho)$. Note that the beforehand given definition is equivalent to the definition of an agreement forest that is usually used in the literature and that we give next. An *agreement forest* $\mathcal{F} = \{F_\rho, F_1, F_2, \ldots, F_k\}$ for S and T is a collection of trees such that the following properties are satisfied:

(i) The label sets $\mathcal{L}(F_\rho), \mathcal{L}(F_1), \mathcal{L}(F_2), \ldots, \mathcal{L}(F_k)$ partition $\mathcal{X} \cup \{\rho\}$ and, in particular, $\rho \in \mathcal{L}(F_\rho)$.

(ii) For each $i \in \{\rho, 1, 2, \ldots, k\}$, we have $F_i \cong S|_{\mathcal{L}(F_i)} \cong T|_{\mathcal{L}(F_i)}$.

(iii) The phylogenetic trees in $\{S(\mathcal{L}(F_i)) \mid i \in \{\rho, 1, 2, \ldots, k\}\}$ and $\{T(\mathcal{L}(F_i)) \mid i = \{\rho, 1, 2, \ldots, k\}\}$ are vertex-disjoint subtrees of S and T, respectively.

Both definitions of an agreement forests for two rooted binary phylogenetic trees are used interchangeably throughout this paper.

An agreement forest with the minimum cardinality among all agreement forests for S and T is called a *maximum-agreement forest* for S and T. An example of an agreement forest for the two trees S and T presented in Figure 1(i), is shown in (ii) of the same figure. It is easy to check that this forest is in fact a maximum-agreement forest for S and T.

A characterization of the hybridization number $h(S,T)$ for two rooted binary phylogenetic trees S and T in terms of agreement forests requires an additional condition. Roughly, this condition avoids that species can inherit genetic material from their own offsprings. Let $\mathcal{F} = \{F_\rho, F_1, F_2, \ldots, F_k\}$ be an agreement forest for two rooted binary phylogenetic $\mathcal{X}$-trees S and T. Furthermore, let $AG(S,T,\mathcal{F})$ be the directed graph whose vertex set is $\mathcal{F}$ and for which (F_i, F_j) is an arc precisely if $i \neq j$, and either

(1) the root of $S(\mathcal{L}(F_i))$ is an ancestor of the root of $S(\mathcal{L}(F_j))$ in S, or

(2) the root of $T(\mathcal{L}(F_i))$ is an ancestor of the root of $T(\mathcal{L}(F_j))$ in T.

We call $\mathcal{F}$ an *acyclic-agreement forest* for S and T if $AG(S,T,\mathcal{F})$ does not contain any directed cycle. To illustrate, Figure 1(iii) shows the graph $AG(S,T,\mathcal{F})$ for S, T, and $\mathcal{F}$ of the same figure. Note that $\mathcal{F}$ is not an acyclic-agreement forest for S and T. Similarly to the definition of a maximum-agreement forest, an acyclic-agreement forest for S and T whose number of components is minimized over all such forests is called a *maximum-acyclic-agreement forest* for S and T. The importance of the concept of acyclic-agreement forests lies in the following theorem that has been established in [**?** , Theorem 2] and gives an attractive characterization of the hybridization number for two rooted binary phylogenetic trees.

Theorem 1. *Let $\mathcal{F} = \{F_\rho, F_1, F_2, \ldots, F_k\}$ be a maximum-acyclic-agreement forest for two rooted binary phylogenetic $\mathcal{X}$-trees S and T. Then*

$$h(S,T) = k.$$

In the proof of Theorem 1, the authors implicitly show that, by deleting all hybridization edges of a hybridization network N that displays two rooted binary phylogenetic $\mathcal{X}$-trees S and T and has a minimum number of hybridization vertices and, subsequently, suppressing all non-root degree-2 vertices, one obtains a maximum-acyclic-agreement forest $\mathcal{F}$ for S and T. Note that $\mathcal{F}$ is well-defined if N is given. We say that N *yields* $\mathcal{F}$. On the other hand, given a maximum-acyclic-agreement forest $\mathcal{F}$ for two rooted binary phylogenetic $\mathcal{X}$-trees S and T, and using the algorithm HybridPhylogeny (for details, see [**?**]) to construct a hybridization network N from $\mathcal{F}$ that displays S and T and yields $\mathcal{F}$, N is rarely unique. Nevertheless, if one aims at reconstructing all hybridization networks that display S and T and whose hybridization number is minimized, one can first calculate all maximum-acyclic-agreement forests for S and T and then construct all possible minimum hybridization networks for each such forest. As mentioned in the introduction, this paper focuses on the first step of this approach, i.e. finding all maximum-acyclic-agreement forests for S and T.

Now, let $\mathcal{F}$ be a set of rooted binary phylogenetic trees, and let a and c be two distinct leaves of $\mathcal{F}$. We say that a and c form a *cherry* in $\mathcal{F}$ if they are adjacent to a common vertex, in which case we denote this cherry by $\{a, c\}$. Note that a and c refer to leaf vertices and not leaf labels. Let

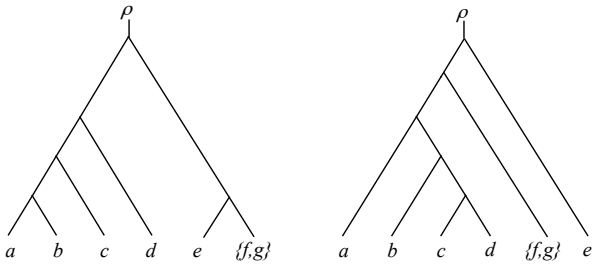


Figure 2: The two phylogenetic trees obtained by calling CHERRYREDUCTION(S, T, $\varnothing$, $\{f, g\}$), where S and T are the two phylogenetic trees shown Figure 1(i).

S and T be two rooted binary phylogenetic $\mathcal{X}$-tree, and let $\mathcal{F}$ be a forest for T. Furthermore, let $\{a, c\}$ be a cherry of $S|_{\mathcal{L}(\overline{\mathcal{F}})}$. We say that $\{a, c\}$ is a *contradicting cherry* of S and $\mathcal{F}$ if there is no cherry $\{a', c'\}$ in $\overline{\mathcal{F}}$ such that one of a' or c', say a', is labeled $\mathcal{L}(a)$ while c' is labeled $\mathcal{L}(c)$. Otherwise, we call $\{a, c\}$ a *common cherry* of S and $\mathcal{F}$.

Cherry reduction. Let $\mathcal{F}$ be a forest for a rooted binary phylogenetic tree, and let $\{a, c\}$ be a cherry of $\mathcal{F}$. The operation of deleting the two leaf vertices a and c and their respective labels and labeling the resulting new leaf vertex with $\mathcal{L}(a) \cup \mathcal{L}(c)$ is called a *cherry reduction*. The new label $\mathcal{L}(a) \cup \mathcal{L}(c)$ is sometimes referred to as a *dummy taxon*. We denote this reduction by $\mathcal{F}[\{\mathcal{L}(a), \mathcal{L}(c)\} \to \mathcal{L}(a) \cup \mathcal{L}(c)]$. Reversely, we denote by $\mathcal{F}[\mathcal{L}(a) \cup \mathcal{L}(c) \to \{\mathcal{L}(a), \mathcal{L}(c)\}]$ the operation of adjoining the vertex labeled $\mathcal{L}(a) \cup \mathcal{L}(c)$ with two new vertices labeled $\mathcal{L}(a)$ and $\mathcal{L}(c)$, respectively, via two new edges and deleting the label $\mathcal{L}(a) \cup \mathcal{L}(c)$. For an example of a cherry reduction, consider the two phylogenetic trees S and T of Figure 1(i) that

have a common cherry $\{f, g\}$. Reducing this cherry in S and T results in the two phylogenetic trees that are shown in Figure 2.

We end this section with an important remark.

Remark 2. *The newly created leaf label, that results from applying a cherry reduction to a cherry $\{a, c\}$ that is common to two rooted phylogenetic trees, is the union of the labels associated with the vertices a and c. For the rest of this paper, we therefore assume that the forest $\mathcal{F}$ before applying a cherry reduction and the forest $\mathcal{F}'$ that results from applying such a reduction have the same label set although the number of leaves has been decreased by one; thus $\mathcal{L}(\mathcal{F}) = \mathcal{L}(\mathcal{F}')$. Furthermore, we write $l(\mathcal{F})$ to denote the number of labeled vertices in $\mathcal{F}$. Clearly, this number is always one greater than the number of leaves in $\mathcal{F}$ due to the vertex labeled ρ. Lastly, let S be a rooted binary phylogenetic tree. We write $l(S) = l(\mathcal{F})$ if the number of labeled vertices in S and $\mathcal{F}$ is identical and if there is a bijection between the vertex labels of S and $\mathcal{F}$.*

3 The algorithm allMAAFs

In this section, we first give a brief outline of the algorithm allMAAFs that calculates all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees and, subsequently, present its pseudocode. Before doing so, we start with an important remark to emphasize how the algorithm presented in this section separates itself from previously published work, and give some additional definitions.

Remark 3. *While* allMAAFs *has a similar flavor as an algorithm presented in [**?**] that has been further improved in [**?**], we remark here that our algorithm contains significant modifications due to a problem in both papers. In particular, Whidden et al.'s algorithms are based on a different definition of an acyclic-agreement forest $\mathcal{F}$ for two rooted binary phylogenetic $\mathcal{X}$-trees S and T compared to the definition that we have given in Section 2. Translated into the language of this paper, they define $\mathcal{F}$ to be acyclic precisely if $AG(S, T, \mathcal{F})$ does not contain a directed cycle of length 2. Of course, this does not eliminate the possibility of cyclic inheritance in general although this is essentially required from a biological point of view. While* allMAAFs *considers this stronger constraint and calculates a maximum-acyclic-agreement forest as defined in Section 2, we additionally show that our algorithm also computes all such forests (see Section 4).*

Let S be a rooted binary phylogenetic $\mathcal{X}$-tree, and let $\mathcal{F}$ be a forest such that $l(S) = l(\overline{\mathcal{F}})$. Let $\{a, c\}$ be a cherry of $S|_{\mathcal{L}(\overline{\mathcal{F}})}$. We denote by e_a the edge of $\mathcal{F}$ that is incident with the leaf vertex, say a', labeled $\mathcal{L}(a)$, and by e_c the edge of $\mathcal{F}$ that is incident with the leaf vertex, say c', labeled $\mathcal{L}(c)$. Furthermore, if $\{a, c\}$ is a contradicting cherry of S and $\mathcal{F}$ and $a \sim_{\mathcal{F}} c$, let F_i be the unique element of $\mathcal{F}$ such that $\mathcal{L}(a) \subset \mathcal{L}(F_i)$ and $\mathcal{L}(c) \subset \mathcal{L}(F_i)$. Let $a', v_1, v_2, \ldots, v_n, c'$ be the path of vertices from a' to c' in F_i. We define $e_B = \{u, v\}$ to be an edge of F_i such that $u \in \{v_1, v_2, \ldots, v_n\}$, $v \notin \{a', v_1, v_2, \ldots, v_n, c'\}$, and u is an ancestor of v in F_i. An example of an edge e_B is shown in Figure 3(i), where e_1 is such an edge for the contradicting cherry $\{a, b\}$ of the two topmost

phylogenetic trees of that figure. Now, an edge e of $\mathcal{F}$ is said to be *associated* with a contradicting or common cherry $\{a, c\}$ for S and $\mathcal{F}$ if one of the following holds:

1. $e \in \{e_a, e_c\}$ if $\{a, c\}$ is a common cherry of S and $\mathcal{F}$, or $\{a, c\}$ is a contradicting cherry of S and $\mathcal{F}$ and $a \not\sim_{\mathcal{F}} c$,

2. $e \in \{e_a, e_B, e_c\}$ if $\{a, c\}$ is a contradicting cherry of S and $\mathcal{F}$ and $a \sim_{\mathcal{F}} c$.

We next describe the pseudocode of allMAAFs. The algorithm takes as input two rooted binary phylogenetic $\mathcal{X}$-trees S and T, a rooted binary phylogenetic tree R and a forest $\mathcal{F}$ such that $l(R) = l(\overline{\mathcal{F}})$ and $\mathcal{L}(T) = \mathcal{L}(\mathcal{F})$, an integer k, and a list M that contains information of previously reduced cherries. The output of allMAAFs is a set $\boldsymbol{\mathcal{F}}$ of forests for $\mathcal{F}$ and an integer k. We will see in Section 4, that if the input to allMAAFs are two rooted binary phylogenetic $\mathcal{X}$-trees S and T, $R = S$, $\mathcal{F} = T$, and $M = \varnothing$, then $\boldsymbol{\mathcal{F}}$ precisely contains all maximum-acyclic-agreement forests for S and T and their respective hybridization number if and only if $k \geq h(S, T)$. We will therefore assume for the rest of the description of the pseudocode that allMAAFs$(S, T, R, \mathcal{F}, k, M)$ has initially been called for $R = S$, $\mathcal{F} = T$, and $M = \varnothing$. If $k < 0$, the algorithm immediately stops and returns an empty set. If, on the other hand, $k \geq 0$ and $l(R) = 0$, then a forest $\mathcal{F}'$ is obtained from $\mathcal{F}$ by calling cherryExpansion$(\mathcal{F}, M)$; that is undoing all previously performed cherry reductions. As we will soon see in Lemma 8, $\mathcal{F}'$ is an agreement forest for S and T. Thus, if the graph $AG(S, T, \mathcal{F}')$ is acyclic, then $\mathcal{F}'$ is an acyclic-agreement-forest for S and T, and the algorithm returns $\mathcal{F}'$ and $|\mathcal{F}'| - 1$ with the latter being the hybridization number for S and T

if $\mathcal{F}$ is of smallest size.

Otherwise, if $k \geq 0$ and $l(R) > 0$, the algorithm proceeds in a bounded-search type fashion by recursively deleting an edge in $\mathcal{F}$ or reducing a common cherry by calling cherryReduction until the resulting forest is a forest for S and T. More precisely, each recursion starts by picking a cherry in R. Since $l(R) > 0$, a cherry, say $\{a, c\}$, always exists since, by definition of $\overline{\mathcal{F}}$, we have $l(R) \geq 2$. Depending on whether $\{a, c\}$ is a contradicting or common cherry of R and $\mathcal{F}$, and on whether or not a and c are vertices of the same component in $\mathcal{F}$, the algorithm branches into at most three computational paths by recursively calling allMAAFs. Note that the number of edge deletions that can additionally be performed at each step of the algorithm is given by the fifth parameter of each call to allMAAFs. In the following, we say that a computational path *corresponds to deleting an edge in* $\mathcal{F}$ if allMAAFs is recursively called for a forest, say $\mathcal{F}'$, that has been obtained from deleting an edge, and $R|_{\mathcal{L}(\overline{\mathcal{F}'})}$. Similarly, we say that a computational path *corresponds to calling* cherryReduction if allMAAFs is recursively called for a tree and a forest that are returned from a call to cherryReduction.

Now, regardless of whether $\{a, c\}$ is a contradicting or common cherry of R and $\mathcal{F}$, allMAAFs branches into two new computational paths that correspond to deleting e_a and e_c in $\mathcal{F}$, respectively. Additionally, if $\{a, c\}$ is a contradicting cherry of R and $\mathcal{F}$ and $a \sim_{\mathcal{F}} c$, then allMAAFs branches into a third computational path that corresponds to deleting an edge e_B in $\mathcal{F}$. Similarly, if $\{a, c\}$ is a common cherry of R and $\mathcal{F}$, then allMAAFs branches into a third path that corresponds to calling cherryReduction$(R, \mathcal{F}, M, \{a, c\})$. Intuitively, if $\{a, c\}$ is a contradicting cherry of R and $\mathcal{F}$, then, to obtain an

agreement forest for the inputted trees S and T, one needs to delete at least one of e_a, e_c and e_B. Otherwise, if $\{a, c\}$ is a common cherry of R and $\mathcal{F}$, then, to obtain an acyclic-agreement forest, say $\mathcal{F}'$ for S and T, either the labels of a and c label vertices of the same component in $\mathcal{F}'$, which is mimicked by calling cherryReduction for a and c, or the labels of a and c are contained in the label sets of two distinct elements in $\mathcal{F}'$; thus one needs to delete one of e_a or e_c. Noting that a common cherry of R and $\mathcal{F}$ is not necessarily a common cherry of S and T, we remark that this part of the algorithm has a similar flavor as [**?** , Lemma 3.1.2], where the authors consider so-called common chains of S and T with at least 3 leaves. The variable k has a central role in our algorithm. Roughly speaking, k is initialized at each recursive call to the minimum value between the value of the variable k passed as parameter and the minimum number of edges we need to cut from $\mathcal{F}$ following the computational path under consideration to obtain an acyclic-agreement forest. So, the variable k tells us the maximal number of edges that we still are allowed to delete from $\mathcal{F}$ following the computational path under consideration to obtain an acyclic-agreement forest of smaller size than the current best. The fact that each forest $\mathcal{G}$ that is returned by a recursive call has size $|\mathcal{G}|$ such that $|\mathcal{G}| + 1 - |\mathcal{F}| = k$ guarantees that only the minimal forests among $\boldsymbol{\mathcal{F}_a}$, $\boldsymbol{\mathcal{F}_c}$, and (depending on the cherry under consideration) $\boldsymbol{\mathcal{F}_B}$ or $\boldsymbol{\mathcal{F}_r}$, are returned to the "parent" recursive call. This both ensures that the algorithm returns the set of all maximum-acyclic-agreement forests when the value of the variable k passed as parameter is greater or equal to $h(S, T)$ and avoids to explore computational paths of the search tree leading to agreement forests having a size greater than the current best

acyclic-agreement forest.

Lines 16-17 avoid to continue exploring the "sibling" paths of a path containing only reductions. Indeed, if such a path has been found, it is pointless to search for a better solution in the sibling paths since they all imply deleting at least one edge.

We end the description of the pseudocode by noting that allMAAFs always terminates because, at each recursive call, either k is decreased by one or the number of leaves in R is decreased by one due to calling cherryReduction.

Algorithm 1: cherryReduction$(R, \mathcal{F}, M, \{a, c\})$

Data: A rooted binary phylogenetic tree R and a forest $\mathcal{F}$ such that $l(R) = l(\overline{\mathcal{F}})$, a list M that contains all information of previously applied cherry reductions, and a common cherry $\{a, c\}$ of R and $\mathcal{F}$.

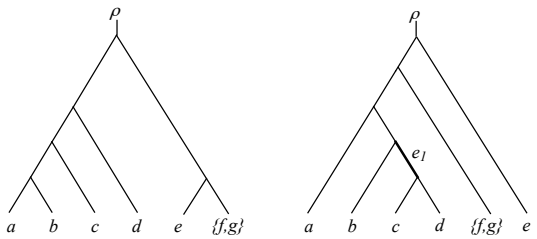
Result: A rooted binary phylogenetic tree R' and a forest $\mathcal{F}'$ obtained from R and $\mathcal{F}$, respectively by replacing $\{a, c\}$ with a single leaf with a new label $\mathcal{L}(a) \cup \mathcal{L}(c)$, and an updated list M'.

1 $M' \leftarrow$ Add $\{\mathcal{L}(a), \mathcal{L}(c)\}$ as last element of M;

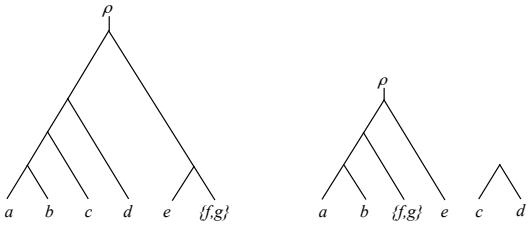
2 $R' \leftarrow R[\{\mathcal{L}(a), \mathcal{L}(c)\} \rightarrow \mathcal{L}(a) \cup \mathcal{L}(c)]$;

3 $\mathcal{F}' \leftarrow \mathcal{F}[\{\mathcal{L}(a), \mathcal{L}(c)\} \rightarrow \mathcal{L}(a) \cup \mathcal{L}(c)]$;

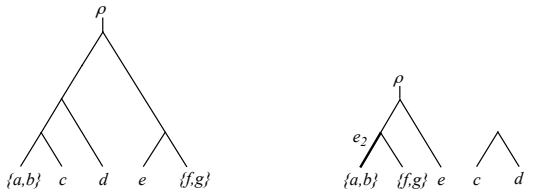
4 **return** $(R', \mathcal{F}', M')$



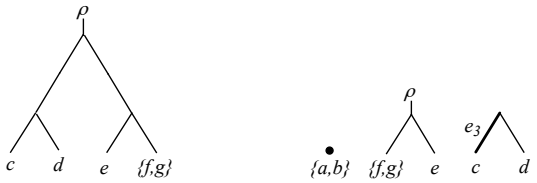
(i)



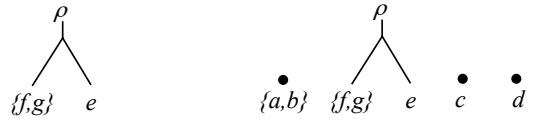
(ii)



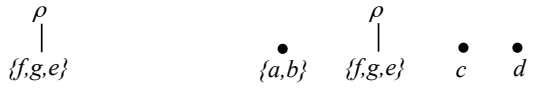
(iii)



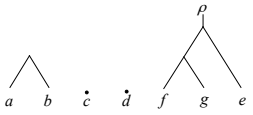
(iv)



(v)



(vi)



(vii)

Figure 3: An example of a call to PROCESSCHERRIES$(S, T, \wedge_1, \wedge_2, \ldots, \wedge_6)$, where S and T are the phylogenetic trees of Figure 1(i) and the cherry list is $((\{f, g\}, \varnothing)$, $(\{a, b\}, e_1)$, $(\{a, b\}, \varnothing)$, $(\{\{a, b\}, c\}, e_2)$, $(\{c, d\}, e_3)$,

Algorithm 2: cherryExpansion$(\mathcal{F}, M)$

Data: A forest $\mathcal{F}$ and a list M containing information of all previously applied cherry reductions.

Result: A forest $\mathcal{F}$ whose vertices labeled with dummy taxa have been replaced by the corresponding cherries using the information contained in M.

```
while M is not empty do
    M ← remove the last element, say {L(a), L(c)}, from M;
    F ← F[L(a) ∪ L(c) → {L(a), L(c)}];
return F
```

4 Correctness of the algorithm allMAAFs

In this section, we prove the main result of this paper. In particular, we show that the algorithm allMAAFs calculates all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees S and T for when inputted with $R = S$, $\mathcal{F} = T$, $M = \varnothing$, and $k \geq h(S, T)$. We start with some additional definitions that are crucial for what follows.

Let $\mathcal{F}$ and $\mathcal{G}$ be two forests such that $\mathcal{L}(\mathcal{F}) = \mathcal{L}(\mathcal{G})$. We call $\mathcal{G}$ a *super-forest* of $\mathcal{F}$ if and only if the following two conditions are satisfied:

(1) for each $G_j \in \mathcal{G}$, there exists a subset $\mathcal{F}'$ of $\mathcal{F}$ such that $\mathcal{L}(\mathcal{F}') = \mathcal{L}(G_j)$, and

(2) for each leaf vertex a in an element of $\mathcal{G}$, there exists a component F_i in $\mathcal{F}$ such that $\mathcal{L}(F_i) \supseteq \mathcal{L}(a)$.

Algorithm 3: allMAAFs(S, T, R, $\mathcal{F}$, k, M)

Data: Two rooted binary phylogenetic $\mathcal{X}$-trees S and T, a rooted binary phylogenetic tree R and a forest $\mathcal{F}$ such that $l(R) = l(\overline{\mathcal{F}})$ and $\mathcal{L}(T) = \mathcal{L}(\mathcal{F})$, an integer k, and a list M that contains information of previously reduced cherries.

Result: A set $\boldsymbol{\mathcal{F}}$ of forests for $\mathcal{F}$ and an integer. In particular, if $\mathcal{F} = T$, $R = S$, $M = \varnothing$, and $k \geq h(S,T)$ is the input to allMAAFs, the output precisely consists of all maximum-acyclic-agreement forests for S and T and their respective hybridization number.

if $k < 0$ **then**
  return $(\varnothing, k-1)$;
if $|l(R)| = 0$ **then**
  $\mathcal{F}' \leftarrow$ cherryExpansion($\mathcal{F}$, M);
  if $AG(S,T,\mathcal{F}')$ is acyclic **then**
    return $(\mathcal{F}', |\mathcal{F}'|-1)$;
  else
    return $(\varnothing, k-1)$;
else
  let $\{a,c\}$ be a cherry of R;
  if $\{a,c\}$ is a common cherry of R and $\mathcal{F}$ **then**
    $(R', \mathcal{F}', M') \leftarrow$ cherryReduction(R, $\mathcal{F}$, M, $\{a,c\}$);
    $(\boldsymbol{\mathcal{F}_r}, k_r) \leftarrow$ allMAAFs(S, T, $R'|_{\mathcal{L}(\overline{\mathcal{F}'})}$, $\mathcal{F}'$, k, M');
    if $\boldsymbol{\mathcal{F}_r} \neq \varnothing$ **then**
      $k \leftarrow \min(k, k_r)$;
    if $k = (|\mathcal{F}|-1)$ **then**
      return $(\boldsymbol{\mathcal{F}_r}, k)$;
  if $k \neq (|\mathcal{F}|-1)$ or $\{a,c\}$ is a contradicting cherry of R and $\mathcal{F}$ **then**
    $(\boldsymbol{\mathcal{F}_a}, k_a) \leftarrow$ allMAAFs(S, T, $R|_{\mathcal{L}(\overline{\mathcal{F}-\{e_a\}})}$, $\mathcal{F}-\{e_a\}$, $k-1$, M);

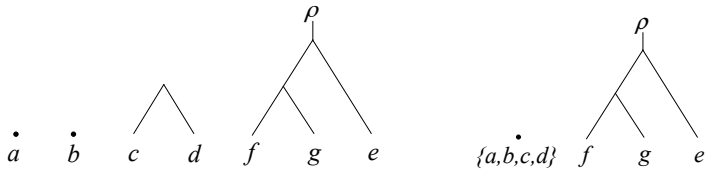


Figure 4: Two non-super-forests of the forest $\mathcal{F}$ in Figure 1(ii). The forest on the left-hand side is not a super-forest of $\mathcal{F}$ because there exists no subset $\mathcal{F}'$ of $\mathcal{F}$ such that $\mathcal{L}(\mathcal{F}') = \{a\}$. The forest on the right-hand side is not a super-forest of $\mathcal{F}$ because there exists no component F_i in $\mathcal{F}$ such that $\mathcal{L}(F_i) \supseteq \{a, b, c\}$.

For an example of two forests that are no super-forests of the forest that is shown in Figure 1(ii), see Figure 4.

The next observation is an immediate consequence of the previous definition.

Observation 1. *Given an acyclic-agreement forest $\mathcal{F}$ for two rooted binary phylogenetic $\mathcal{X}$-trees S and T, then S and T are both super-forests for $\mathcal{F}$.*

Let R be a rooted binary phylogenetic tree and let $\mathcal{F}$ be a forest such that $l(R) = l(\overline{\mathcal{F}})$. Furthermore, let $\{a, c\}$ be a cherry of R. In the following, we say that a pair $\wedge = (\{a, c\}, e)$ is a *cherry action* if one of the following conditions is satisfied:

(1) $\{a, c\}$ is a contradicting or common cherry of R and $\mathcal{F}$ and e is an edge associated with $\{a, c\}$, or

(2) $\{a, c\}$ is a common cherry of R and $\mathcal{F}$ and $e = \varnothing$.

Finally, we say that $\Lambda = (\wedge_1, \wedge_2, \ldots \wedge_l)$ is a *cherry list* for R and $\mathcal{F}$ if and only

if each $\wedge_i$ is a cherry action in iteration i of the following algorithm; i.e. processCherries does not return *false*:

processCherries$(R, \mathcal{F}, (\wedge_1, \wedge_2, \ldots, \wedge_l))$

$M \leftarrow \varnothing$;

for each $i = 1, \ldots, l$

$(\{a, c\}, e_i) \leftarrow \wedge_i$;

if $\{a, c\}$ is a common cherry of R and $\mathcal{F}$ and $e_i = \varnothing$

$(R, \mathcal{F}, M) \leftarrow$ cherryReduction$(R, \mathcal{F}, M, \{a, c\})$;

else if $\{a, c\}$ is a common or contradicting cherry of R and $\mathcal{F}$ and e_i is associated with $\{a, c\}$

$\mathcal{F} \leftarrow \mathcal{F} - \{e_i\}$;

$R \leftarrow R|_{\mathcal{L}(\overline{\mathcal{F}})}$;

else

return (*false*);

$\mathcal{F} \leftarrow$ cherryExpansion$(\mathcal{F}, M)$;

return $(R, \mathcal{F}, M)$;

Remark 4. *The algorithm* processCherries *is mimicking a computational path of the algorithm* allMAAFs *for when the former algorithm is given a cherry list for R and $\mathcal{F}$. A specific example of a call to* processCherries *is shown in Figure 3 with a detailed description given in the caption of this figure.*

In what follows, we will sometimes make use of the algorithm processCherries$(R, \mathcal{F}, \Lambda)$, but without executing the call to cherryExpansion in the second-to-last line of this algorithm. We refer to this slightly

different algorithm as processCherries$^*(R, \mathcal{F}, \Lambda)$ and to the returned forest as a *reduced forest*. Now, let $\mathcal{F}'$ be the forest obtained from calling processCherries$(R, \mathcal{F}, \Lambda)$, and let $\mathcal{F}''$ be the forest obtained from calling processCherries$^*(R, \mathcal{F}, \Lambda)$. We say that $\mathcal{F}'$ is the *underlying forest for* $\mathcal{F}''$ and observe that $|\mathcal{F}'| = |\mathcal{F}''|$.

We continue with two important remarks.

Remark 5. *Applying* processCherries$^*(R, \mathcal{F}, \Lambda)$*, returns a tree* R *that does not contain any vertex if and only if, prior to calling* cherryExpansion$(\mathcal{F}, M)$*, the forest* $\mathcal{F}$ *only consists of isolated vertices and possibly an element that precisely contains a vertex labeled* ρ *that is attached to a vertex by an edge; i.e.* $\overline{\mathcal{F}} = \varnothing$ *(for an example, see Figure 3(vi)).*

Remark 6. *By the definition of* $\overline{\mathcal{F}}$*, note that applying* processCherries* *never returns a tree* R *that consists of a single leaf attached to the root vertex labeled* ρ*.*

Now, let $\mathcal{G}$ and $\mathcal{F}$ be two forests such that $\mathcal{G}$ is a super-forest of $\mathcal{F}$. Furthermore, let e be an edge and $\{a, c\}$ be a cherry (if it exists) of $\mathcal{G}$. We say that e *is a bad choice* for $\mathcal{G}$ and $\mathcal{F}$ if $\mathcal{G} - \{e\}$ is not a super-forest of $\mathcal{F}$. Note that $\mathcal{G} - \{e\}$ always satisfies Condition (2) in the definition of a super-forest. Similarly, we say that $\{a, c\}$ *is a bad choice* for $\mathcal{G}$ and $\mathcal{F}$ if the forest, say $\mathcal{G}'$, that is obtained from $\mathcal{G}$ by reducing the cherry $\{a, c\}$ to a new leaf is not a super-forest of $\mathcal{F}$. Note that $\mathcal{G}'$ always satisfies Condition (1) in the definition of a super-forest.

We next prove two lemmas that are necessary to establish the main result (Theorem 9) of this paper.

Lemma 7. *Let Λ be a cherry list for two rooted binary phylogenetic $\mathcal{X}$-trees S and T, and let $\mathcal{F}$ be a maximum-acyclic-agreement forest for S and T. Additionally, let S' and $\mathcal{G}'$ be the tree and the forest, respectively, that have been obtained from calling* processCherries*$^*(S, T, \Lambda)$, *and let $\mathcal{G}$ be the underlying forest for $\mathcal{G}'$. If $\mathcal{G}'$ is a super-forest for $\mathcal{F}$, then S' contains at least one cherry or $\mathcal{G} = \mathcal{F}$.*

Proof. Suppose that this is not true. Thus, $l(S') = 0$ (see Remark 6) and there exists an element in $\mathcal{F}$ that is not an element in $\mathcal{G}$. Furthermore, since both forests are forests of T, we cannot have that there exists $F_i \in \mathcal{F}$ and $G_j \in \mathcal{G}$ such that $\mathcal{L}(F_i) = \mathcal{L}(G_j)$ and $F_i \notin G_j$. Since $\mathcal{G}'$ is a super-forest for $\mathcal{F}$ other than $\mathcal{F}$, there exist at least two components F_i and F_j of $\mathcal{F}$ such that $\mathcal{L}(F_i) \cup \mathcal{L}(F_j) \subseteq \mathcal{L}(G_k)$, where G_k is an element of $\mathcal{G}'$. Furthermore, since $l(S') = 0$, we have $\overline{\mathcal{G}'} = \varnothing$ (see Remark 5). Now, since $G_k \in \mathcal{G}'$, either G_k is an isolated vertex a such that $\mathcal{L}(a) = \mathcal{L}(G_k) \supseteq (\mathcal{L}(F_i) \cup \mathcal{L}(F_j))$, or G_k is a leaf vertex that is attached to the vertex labeled ρ by an edge such that $\mathcal{L}(a) \cup \{\rho\} = \mathcal{L}(G_k) \supseteq (\mathcal{L}(F_i) \cup \mathcal{L}(F_j))$. Since neither $\mathcal{L}(F_i) = \{\rho\}$ nor $\mathcal{L}(F_j) = \{\rho\}$ (see [**?** , Lemma 1]), $\mathcal{G}'$ does not fulfill Condition (2) in the definition of a super-forest; a contradiction. □

Lemma 8. *Let S and T be two rooted binary phylogenetic $\mathcal{X}$-trees, and let $\mathcal{F}$ be a forest that is returned from calling* cherryExpansion *(line 4 of the pseudocode of Algorithm 3) while executing* allMAAFs$(S, T, S, T, k, \varnothing)$. *Then, $\mathcal{F}$ is an agreement forest for S and T.*

Proof. Let $\mathcal{F}'$ be the forest for which calling cherryExpansion returns $\mathcal{F}$. By construction, $\mathcal{F}$ is a forest for T. Now, for the purpose of deriving a contradiction, assume that $\mathcal{F}$ is not a forest for S. Since $\mathcal{F}$ is a forest for T and $\mathcal{L}(S) = \mathcal{L}(T)$, the label sets of the elements in $\mathcal{F}$ partition $\mathcal{L}(S)$. Thus, it is sufficient to consider the following two cases:

Case (1). Assume that there exists an element F_i in $\mathcal{F}$ such that $S|_{\mathcal{L}(F_i)} \not\cong F_i$. Since $l(R) = 0$ (line 3 of the pseudocode of Algorithm 3), we have by Remark 5 that $\overline{\mathcal{F}'} = \varnothing$. This implies that the element of $\mathcal{F}$ with leaf sets $\mathcal{L}(F_i)$ has been shrunk to a single vertex or to a single leaf that is attached to the vertex labeled ρ by an edge. But, since $S|_{\mathcal{L}(F_i)} \not\cong F_i$, one of the cherry reductions that has been used to shrink $T|_{\mathcal{L}(F_i)}$ is called for a cherry that is not a cherry of R, where R is the tree that is considered in some recursive call of allMAAFs$(S, T, S, T, k, \varnothing)$ (see pseudocode of Algorithm 3); a contradiction.

Case (2). Assume that there exist two elements F_i and F_j in $\mathcal{F}$ such that $S(\mathcal{L}(F_i))$ and $S(\mathcal{L}(F_j))$ are not vertex-disjoint in S. Let $S' = S|_{\mathcal{L}(F_i) \cup \mathcal{L}(F_j)}$. For example, the simplest case is shown in Figure 5, where the subtrees in white are part of F_i and the ones in black of F_j. In general, a straightforward check now shows that it is not possible to shrink both $T|_{\mathcal{L}(F_i)}$ and $T|_{\mathcal{L}(F_j)}$ to two distinct single vertices in $\mathcal{F}'$ (one possibly being attached to the vertex labeled ρ) by using cherry reductions because to shrink one of the two components to a single vertex it is necessary to cut a subtree of the other component, thereby contradicting that F_i and F_j are both elements in $\mathcal{F}$.

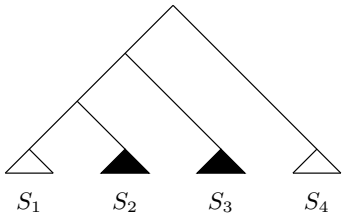


Figure 5: An example of a rooted phylogenetic tree S' that is used in Case (2) of the proof of Lemma 8 (for details, see text).

Referring back to Figure 5, F_j cannot be shrunk to a single vertex by using a list of cherry reductions without cutting S_1.

Combining both cases establishes the lemma. □

Theorem 9. *Let S and T be two rooted binary phylogenetic $\mathcal{X}$-trees. Calling*

$$\text{allMAAFs}(S, T, S, T, k, \varnothing)$$

returns all maximum-acyclic-agreement forests for S and T if and only if $k \geq h(S, T)$.

Proof. By Lemma 8, each forest that is calculated in the course of executing allMAAFs$(S, T, S, T, k, \varnothing)$ and checked for acyclicity (see line 5 of the pseudocode of Algorithm 3) is an agreement forest for S and T. Thus, if $k \geq h(S, T)$, each forest that is returned from running the algorithm is an acyclic-agreement forest for S and T. Moreover, since k is updated to take advantage of the size of the best solutions that previous recursive calls have found (lines 15, 21, 24 and 34), only maximum-acyclic-agreement forests are returned. It is therefore sufficient to show that each maximum-acyclic-agreement forest for S and T is returned by the algorithm.

Let allMAAFs$(S, T, S, T, k, \varnothing)$ be a call of Algorithm 3, and, for each $l \in \{1, 2, \ldots, h(S,T)+1\}$, let $\mathcal{G}_l$ be the set of all reduced forests of size l that have been computed by executing this call. In other words, $\mathcal{G}_l$ precisely contains all forests that are used as a parameter in a recursive call to allMAAFs in lines 13, 19, 22 and 32 of the pseudocode and, in particular, $T \in \mathcal{G}_1$. Furthermore, let $\mathcal{F}$ be a maximum-acyclic-agreement forest for S and T. We will prove that, for each $l \in \{1, 2, \ldots, h(S,T) + 1\}$, the set $\mathcal{G}_l$ contains a reduced forest $\mathcal{G}'$ that is a super-forest for $\mathcal{F}$. This implies that $\mathcal{G}_{h(S,T)+1}$ contains a reduced forest $\mathcal{G}'$ that is a super-forest of $\mathcal{F}$ such that $|\mathcal{F}| = |\mathcal{G}|$, where $\mathcal{G}$ is the underlying forest of $\mathcal{G}'$. Hence, as $\mathcal{G}$ and $\mathcal{F}$ are both forests for T, it follows that $\mathcal{G}$ is isomorphic to $\mathcal{F}$, thereby establishing the theorem.

We proceed by induction on l. If $l = 1$, then the result follows from Observation 1 and because $T \in \mathcal{G}_1$. Now suppose that the result holds whenever $l \leq h(S,T)$. We will next show that the claim holds for $l+1$. Let $\mathcal{G}'$ be a reduced forest of size l such that $\mathcal{G}'$ is a super-forest for $\mathcal{F}$. By the induction assumption, $\mathcal{G}'$ exists. Let $\mathcal{G}$ be the underlying forest of $\mathcal{G}'$. Furthermore, let $\Lambda^{\mathcal{G}}$ be the cherry list that has been used by calling allMAAFs$(S, T, S, T, k, \varnothing)$ to construct $\mathcal{G}'$, and let R be the phylogenetic tree that is returned from calling processCherries*$(S, T, \Lambda^{\mathcal{G}})$. Since $|\mathcal{G}| < |\mathcal{F}|$, it follows from Lemma 7 that R contains a cherry $\{a, c\}$. Let $\mathcal{L}(a) \subset \mathcal{X}$ and $\mathcal{L}(c) \subset \mathcal{X}$ be the label sets of the leaf vertices a and c, respectively, in R, and let $a' \in \mathcal{L}(a)$ and $c' \in \mathcal{L}(c)$. Furthermore, if $\{a, c\}$ is a contradicting cherry for R and $\mathcal{G}'$ and $a \sim_{\mathcal{G}'} c$, let $\mathcal{L}(B) \subset \mathcal{X}$ be the union of labels of all leaf vertices that are contained in the pendant subtree below e_B in $\mathcal{G}'$. Note that, since $\mathcal{G}'$ is a super-forest for $\mathcal{F}$, we have that there exist two elements $F_i, F_j \in \mathcal{F}$, not necessarily distinct,

such that $\mathcal{L}(a) \subseteq \mathcal{L}(F_i)$ and $\mathcal{L}(c) \subseteq \mathcal{L}(F_j)$. The rest of the proof distinguishes two cases depending on whether $\{a, c\}$ is a contradicting or common cherry for R and $\mathcal{G}'$.

First, suppose that $\{a, c\}$ is a contradicting cherry for R and $\mathcal{G}'$. To derive a contradiction, assume that $\mathcal{G}_{l+1}$ does not contain any reduced forest that is a super-forest of $\mathcal{F}$. In particular, this implies that deleting any edge associated with $\{a, c\}$ is a bad choice for $\mathcal{G}'$ and $\mathcal{F}$ since no resulting forest is a super-forest for $\mathcal{F}$ although they all satisfy Condition (2) in the definition of a super-forest. Thus, one of the following holds:

(1) $a \nsim_{\mathcal{G}'} c$ and both edges $\{e_a\}$ and $\{e_c\}$ are bad choices for $\mathcal{G}'$ and $\mathcal{F}$;

(2) $a \sim_{\mathcal{G}'} c$ and all edges $\{e_a\}$, $\{e_B\}$ and $\{e_c\}$ are bad choices for $\mathcal{G}'$ and $\mathcal{F}$.

Case (1). Observe that neither $\mathcal{G}' - \{e_a\}$ nor $\mathcal{G}' - \{e_c\}$ is a super-forest of $\mathcal{F}$. This implies that $\mathcal{F}$ does not contain an element F_i such that $\mathcal{L}(a) = \mathcal{L}(F_i)$ or $\mathcal{L}(c) = \mathcal{L}(F_i)$. Thus $\mathcal{F}$ contains two distinct components F_j and F_k such that $\mathcal{L}(a) \subset \mathcal{L}(F_j)$ and $\mathcal{L}(c) \subset \mathcal{L}(F_k)$, and for which there exist elements $x, y \in \mathcal{X}$ such that $x \in \mathcal{L}(F_j)$, $x \notin \mathcal{L}(a)$, $y \in \mathcal{L}(F_k)$, and $y \notin \mathcal{L}(c)$. By construction, each of x and y is contained in a label of a distinct leaf in R. Now, recalling that $\{a, c\}$ is a cherry of R, we have that $\mathrm{lca}_R(a', c', x, y)$ is an ancestor of $\mathrm{lca}_R(a', c')$ and, therefore, $\mathrm{lca}_S(a', c', x, y)$ is an ancestor of $\mathrm{lca}_S(a', c')$. Furthermore, as $a', x \in \mathcal{L}(F_j)$ and $c', y \in \mathcal{L}(F_k)$, it now follows that $S(\mathcal{L}(F_j))$ and $S(\mathcal{L}(F_k))$ do both have the vertex $\mathrm{lca}_S(a', c')$ in common; thereby contradicting that $\mathcal{F}$ is an agreement forest for S and T.

Case (2). Observe that no forest in $\{\mathcal{G}' - \{e_a\}, \mathcal{G}' - \{e_B\}, \mathcal{G}' - \{e_c\}\}$ is a super-forest of $\mathcal{F}$. This implies that $\mathcal{F}$ does not contain any element F_i such that $\mathcal{L}(a) = \mathcal{L}(F_i)$ or $\mathcal{L}(c) = \mathcal{L}(F_i)$ or a subset $\mathcal{F}'$ of $\mathcal{F}$ such that $\mathcal{L}(B) = \mathcal{L}(\mathcal{F}')$. We next consider three subcases.

First, assume that $\mathcal{F}$ contains a component F_j such that $\mathcal{L}(a) \subset \mathcal{L}(F_j)$, $\mathcal{L}(c) \subset \mathcal{L}(F_j)$ and there exists at least one element in the intersection $\mathcal{L}(B) \cap \mathcal{L}(F_j)$. Let b' be such an element. By construction, each of a', b', and c' is contained in a label of a distinct leaf in R. Now, recalling that $\{a, c\}$ is a cherry of R, we have that $\mathrm{lca}_R(a', b', c')$ is an ancestor of $\mathrm{lca}_R(a', c')$ and, therefore, $\mathrm{lca}_S(a', b', c')$ is an ancestor of $\mathrm{lca}_S(a', c')$. On the contrary, let G_k be the element of $\mathcal{G}'$ that contains the leaf labeled $\mathcal{L}(a)$ and the leaf labeled $\mathcal{L}(c)$. Since G_k also contains a leaf whose label contains b' and due to the definition of e_B, we have that $\mathrm{lca}_{G_k}(a', b', c')$ is an ancestor of $\mathrm{lca}_{G_k}(a', b')$ or $\mathrm{lca}_{G_k}(c', b')$ and, therefore, $\mathrm{lca}_T(a', b', c')$ is an ancestor of $\mathrm{lca}_T(a', b')$ or $\mathrm{lca}_T(c', b')$. Thus, $S|\{a', b', c'\} \ncong T|\{a', b', c'\}$; thereby contradicting that $\mathcal{F}$ is a maximum-acyclic-agreement forest for S and T.

Second, assume that $\mathcal{F}$ contains a component F_j such that $\mathcal{L}(a) \subset \mathcal{L}(F_j)$, $\mathcal{L}(c) \subset \mathcal{L}(F_j)$, and $\mathcal{L}(B) \cap \mathcal{L}(F_j) = \varnothing$. Then, since there exists no subset $\mathcal{F}'$ of $\mathcal{F}$ such that $\mathcal{L}(B) = \mathcal{L}(\mathcal{F}')$, there exists a distinct element $F_k \in \mathcal{F}$ such that $b' \in \mathcal{L}(F_k)$ for any $b' \in \mathcal{L}(B)$ and there exists an element $x \in \mathcal{X}$ for which $x \in \mathcal{L}(F_k)$ and $x \notin \mathcal{L}(B)$. Let G_k be the element of $\mathcal{G}'$ that contains the leaf labeled $\mathcal{L}(a)$. Clearly, G_k also contains the leaf labeled $\mathcal{L}(c)$ and the leaf whose label contains b'. Furthermore, since $\mathcal{G}$ is a super-forest for $\mathcal{F}$, note that G_k contains a leaf whose label contains x. Furthermore, by the definition of e_B, we have that the $\mathrm{lca}_{G_k}(b', x)$ lies on the path from the leaf

labeled a' to the leaf labeled c' in G_k and, therefore, the $\mathrm{lca}_T(b', x)$ lies on the path from the leaf labeled $\mathcal{L}(a)$ to the leaf labeled $\mathcal{L}(c)$ in T. Now, it is easily checked that F_j and F_k are not vertex-disjoint in T; thereby contradicting that $\mathcal{F}$ is an agreement forest for S and T.

Third, assume that $\mathcal{F}$ contains two components F_j and F_k such that $\mathcal{L}(a) \subset \mathcal{L}(F_j)$ and $\mathcal{L}(c) \subset \mathcal{L}(F_k)$. Hence, there exist elements $x, y \in \mathcal{X}$ such that $x \in \mathcal{L}(F_j)$, $x \notin \mathcal{L}(a)$, $y \in \mathcal{L}(F_k)$, and $y \notin \mathcal{L}(c)$. Note that x or y may or may not be elements of $\mathcal{L}(B)$. In this case, it is straightforward to see that we can derive the same contradiction as in Case (1).

By combining Cases (1) and (2), we deduce that there exists a super-forest of $\mathcal{F}$ that can be constructed from $\mathcal{G}'$ by deleting one of $\{e_a, e_c\}$ if $a \nsim_{\mathcal{G}'} c$ or one of $\{e_a, e_B, e_c\}$ if $a \sim_{\mathcal{G}'} c$. Thus, this super-forest is an element of $\mathcal{G}_{l+1}$.

Second, suppose that $\{a, c\}$ is a common cherry for R and $\mathcal{G}'$. Again, to derive a contradiction, assume that $\mathcal{G}_{l+1}$ does not contain any reduced forest that is a super-forest of $\mathcal{F}$. In particular, this implies that e_a, e_c, and $\{a, c\}$ are all bad choices for $\mathcal{G}'$ and $\mathcal{F}$. Thus, similar to Case (1), $\mathcal{F}$ contains two distinct components F_j and F_k such that $\mathcal{L}(a) \subset \mathcal{L}(F_j)$ and $\mathcal{L}(c) \subset \mathcal{L}(F_k)$, and for which there exist elements $x, y \in \mathcal{X}$ such that $x \in \mathcal{L}(F_j)$, $x \notin \mathcal{L}(a)$, $y \in \mathcal{L}(F_k)$, and $y \notin \mathcal{L}(c)$. Applying the same argument as in Case (1), this contradicts the fact that the elements of $\mathcal{F}$ are vertex-disjoint in S. Thus, one of e_a, e_c, or $\{a, c\}$ is not a bad choice for $\mathcal{G}'$ and $\mathcal{F}$. If e_a or e_c is not a bad choice for $\mathcal{G}'$ and $\mathcal{F}$, then $\mathcal{G}_{l+1}$ clearly contains a forest that is a super-forest for $\mathcal{F}$.

On the other hand, if e_a and e_c are both bad choices for $\mathcal{G}'$ and $\mathcal{F}$, then $\{a, c\}$ is not such a choice. Hence, calling cherryReduction$(R, \mathcal{G}', M, \{a, c\})$ returns a forest $\mathcal{G}''$ that is a super-forest for $\mathcal{F}$. Note that the underlying forest of $\mathcal{G}''$ is $\mathcal{G}$. Since $|\mathcal{G}| < |\mathcal{F}|$, it follows from Lemma 7, that after some additional recursions of allMAAFs, the algorithm chooses a cherry in line 10 of the pseudocode of allMAAFs and subsequently deletes an edge in order to obtain a forest of size $|\mathcal{G}| + 1$. Then by applying the arguments of Cases (1) and (2), and the argument of this paragraph (depending on the type of cherry the algorithm has chosen), it is easily checked that $\mathcal{G}_{l+1}$ contains a forest that is a super-forest of $\mathcal{F}$. This completes the proof of the theorem. □

5 Running time of the algorithm

In this section, we detail the running time of the algorithm allMAAFs.

Theorem 10. *Let S and T be two rooted binary phylogenetic $\mathcal{X}$-trees, and let k be an integer. The running time of* allMAAFs$(S, T, S, T, k, \varnothing)$ *is* $O(3^{|\mathcal{X}|})$.

Proof. Let $\mathcal{F}$ be a forest for T that has been obtained from T by deleting n edges. Recall that allMAAFs stops when $\overline{\mathcal{F}} = \varnothing$ (see Remark 5). It is easy to see that, $|\mathcal{X}| - n - 1$ cherry reductions are needed to reduce $\mathcal{F}$ to a forest, say $\mathcal{F}'$, such that $\overline{\mathcal{F}'} = \varnothing$. Thus, the number of recursive calls is $O(|\mathcal{X}|)$. Since allMAAFs is called for at most 3 times from within each recursion, it now follows that the running time of allMAAFs$(S, T, R, \mathcal{F}, k, M)$ is $O(3^{|\mathcal{X}|})$ as claimed. □

While the worst-case running time that is presented in Theorem 10 is

purely theoretical, it can be significantly optimized in the following way. Bordewich and Semple [**?**] showed that the problem of calculating the minimum number of hybridization events that is needed to simultaneously explain two rooted binary phylogenetic $\mathcal{X}$-trees S and T is fixed-parameter tractable. They used two reductions – called the *subtree and chain reduction* – to establish this result. Loosely speaking, these reductions replace different types of features that are common to S and T with a small number of new leaves, thereby shrinking the original trees to their respective cores while preserving their hybridization number in a well-defined way. In fact, these two reductions are sufficient to yield a kernelization of the above-mentioned problem. More precisely, it is shown in [**?** , Lemma 3.3] that, by repeatedly applying the subtree and chain reductions to S and T until no further reduction is possible, the leaf set size of the so-obtained rooted binary phylogenetic trees is at most $14h(S,T)$. It is now straightforward to see that modifying allMAAFs$(S,T,R,\mathcal{F},k,M)$ in the following way is sufficient to make use of this result.

1. If $R = S$ and $\mathcal{F} = T$, apply the subtree and chain reduction until no further reduction is possible and directly return $(\varnothing, k-1)$ if the leaf set size of the obtained trees is greater than $14k$.

2. Introduce a new global variable, say w, that is used to keep track of the weight of each initially reduced common chain of S and T (for details, see [**?**]). Additionally, whenever cherryExpansion is called for a forest throughout a run of allMAAFs, also call subtreeExpansion and chainExpansion to reverse each initially performed subtree and

chain reduction, respectively.

3. For each potential acyclic-agreement forest $\mathcal{F}'$ for S and T that is returned from calling cherryExpansion, subtreeExpansion, and chainExpansion (see line 4 of the pseudocode of allMAAFs), do not only check if $\mathcal{F}'$ is acyclic, but also whether or not it is a so-called legitimate-agreement forest (for details, see [**?**]). Note that this additional check can be performed in polynomial time.

We denote this extended version by allMAAFs*$(S, T, S, T, k, \varnothing)$.

Now, noting that the subtree and chain reduction can be computed in $O(n^3)$ for two rooted binary phylogenetic $\mathcal{X}$-trees, where $n = |\mathcal{X}|$ [**?**], the next corollary is an immediate consequence of Theorem 10, and the kernelization ideas that are presented in [**?**] and briefly summarized prior to this paragraph.

Corollary 11. *Let S and T be two rooted binary phylogenetic $\mathcal{X}$-trees, and let k be an integer. The running time of* allMAAFs*$(S, T, S, T, k, \varnothing)$ *is* $O(3^{14k} + n^3)$*, where* $n = |X|$.

We next outline why, despite the theoretical worst-case running time, the practical running time of allMAAFs is quick.

Practical running time An alternative approach to calculating all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees is to delete all possible subsets of edges in the first tree and to check which of the resulting forests are acyclic. If one considers the forests of the first tree by

increasing number of components and stops as soon as one finds an acyclic-agreement forest whose size is greater than the smallest acyclic-agreement forest found, this yields an algorithm whose theoretical worst-case running time is less than the worst-case running time of allMAAFs. An algorithm that uses advanced ideas of this approach and calculates one maximum-acyclic-agreement forest was presented in [**?**]. However, subsequently to the publication of [**?**], two algorithms [**?** **?**] have been published that outperform the former algorithm for many data sets. Moreover, preliminary results (published in the third author's Master's thesis [**?**]) indicate that extending the algorithm of [**?**] in a way so that it finds all maximum-acyclic-agreement forests results in an algorithm that is slower than allMAAFs in terms of practical running times.

To get an idea why our algorithm in practice performs better than such a naive algorithm having a better worst-case running time, one has to consider the following issues with regards to allMAAFs:

1. Many computational paths of the search tree are not considered in their full depth due to the use of k (this is particularly important if $k > h(S,T)$) and to the presence of lines 16-17.

2. The theoretical running time presented in Theorem 3 considers the worst-case scenario which is achieved by alternatingly processing contradicting and common cherries. The best case scenario, in contrast, is achieved when only contradicting cherries are processed during the first k recursive calls of each search path. Since only forests of size smaller or equal to k are considered, our search tree has at most $O(3^k)$ different

leaves in this case and thus, at most $O(3^k)$ recursive calls are performed in total. This means there is a large gap between the worst-case and the best-case running times. By applying the subtree and chain reduction before entering the algorithm one can reduce the number of possible cherries to process what pushes the practical running time towards the running time of the best case scenario.

Nevertheless, if $h(S,T)$ is sufficiently small, a search and bound approach considering all possible solutions for increasing values of k may be faster than our approach. In summary, we think that the practical running time of allMAAFs is competitive, which may be an important reason for many biologists to use our algorithm in order to analyze their data sets.

6 Discussion

A topical question in current mathematical research on reticulate evolution is how to construct *all* rooted hybridization networks that display a pair of rooted binary phylogenetic trees such that the number of hybridization vertices is minimized. In this paper, we have made a first step towards achieving this goal by developing the first non-naive algorithm—called allMAAFs—that computes all maximum-acyclic-agreement forests for two rooted binary phylogenetic trees on the same taxa set. Furthermore, we have shown that the algorithm presented in [**?**], which is freely available as part of Dendroscope [**?**], is correct by providing formal proofs. Despite the theoretical worst-case running time of allMAAFs, the authors of [**?**] have shown that it runs quickly for many biological and simulated data sets. At the end of

Section 5 we gave some reasons why the practical running time is good. It is part of ongoing research to extend the algorithm HybridPhylogeny [**?**] in order to compute *all* possible hybridization networks that display a pair of rooted phylogenetic trees and whose number of hybridization vertices is minimized when a maximum-acyclic-agreement forest for these two trees given. In combination with allMAAFs, such an algorithm will then compute all possible minimum hybridization networks that display a pair of phylogenetic trees.

Acknowledgements

We would like to thank Daniel Huson for helpful discussions and two anonymous reviewers for their comments. Financial support from the University of Tübingen is gratefully acknowledged. This work has been partially funded by the ANCESTROME project ANR-10-IABI-0-01. This publication is the contribution no. 2012-XXX of the Institut des Sciences de l'Evolution de Montpellier (UMR 5554-CNRS-IRD).