Autumn Algorithm – Computation of Hybridization Networks for Realistic Phylogenetic Trees

Daniel H. Huson and Simone Linz

Abstract—A minimum hybridization network is a rooted phylogenetic network that displays (or contains) two given rooted phylogenetic trees using a minimum number of reticulations. Much mathematical work has been done on the calculation of such networks, usually making simplifying assumptions on the input trees, such as requiring them to be bifurcating, correctly rooted or that they both contain the same taxa. In biological studies, these assumptions usually do not hold and "realistic" trees have multifurcations (obtained by contracting uncertain edges), are difficult to root and rarely contain exactly the same taxa (due to absent genes or missing data). In this paper, we present a new algorithm for computing minimum hybridization networks for a given pair of "realistic" rooted phylogenetic trees that are not necessarily bifurcating and do not necessarily have exactly the same taxa. We also describe how the algorithm might be used to improve the rooting of the input trees. We introduce the concept of "autumn trees", which provides a nice framework for the formulation of algorithms based on the mathematics of "maximum acyclic agreement forests". While the main computational problem addressed in this paper is hard, the run-time depends mainly on how different the given input trees are. In biological studies, where the input trees are reasonably similar, our parallel implementation performs well in practice. We have implemented the algorithm in our open source program Dendroscope 3, which provides an ideal platform for biologists to explore the use of rooted phylogenetic networks to represent their data. We demonstrate the utility of the algorithm by applying it to several previously studied data sets.

Index Terms—Computational biology, phylogenetics, networks, hybridization, algorithms

1 Introduction

The evolutionary history of a set of species is usually represented by a rooted phylogenetic tree, in which leaves represent extant species, internal nodes represent speciation events and the root node represents the last common ancestor of the species under study [1], [2]. Unrooted phylogenetic networks are commonly used to represent incompatibilities in data or conflicting evolutionary histories [3]. Rooted phylogenetic networks may potentially be employed to represent evolutionary histories in which reticulations such as hybridization or horizontal gene transfer are believed to play a significant role [4].

While *unrooted* phylogenetic networks are frequently used in publications (as is evident by the good number of citations of papers that describe such approaches [3], [5], [6]), *rooted* phylogenetic networks appear much less often and usually the presented networks are constructed semi-automatically, (see e.g. [7], [8]). One impediment against the use of rooted phylogenetic networks in evolutionary studies has been the lack of appropriate tools for computing them.

In this paper we focus on the problem of computing a *hybridization network* for a given pair of rooted phylogenetic tree. Previously published methods that address this problem, or the closely related problem of computing the hybridization number, make the mathematically simplifying assumptions that the two input trees are bifurcating (i.e. fully resolved), or that they both contain exactly the same taxa [9]–[14]. There exist several algorithms to compute an optimal hybridization network for two bifurcating trees on the same taxon set [12], [15]. Moreover, an algorithm exists for computing the hybridization network for two multifurcating trees [16], which uses a bounded-search-type approach and is based on ideas from the softwired clusters literature [22]. Further, for the closely related problem of finding the rooted subtree prune and regraft (rSPR) distance between two rooted trees, two algorithms [17], [18] have recently been developed that can handle multifurcating trees. However, there seems to be no straightforward modification of the algorithms such that they can compute a minimum hybridization network for two multifurcating trees.

In practice, *realistic* phylogenetic trees considered in biological studies are usually not bifurcating and seldom contain exactly the same taxa. Moreover, the assumption that the trees are correctly rooted is often not fulfilled. Here, we propose a new method that takes as input two rooted phylogenetic trees, not necessarily bifurcating, on overlapping, but not necessarily identical taxon sets, and computes a "representative set" of all minimum hybridization networks that contain both trees. Our main goal is to overcome the practical limitations of previously published methods.

In addition, we provide an implementation of our algorithm that aims at determining a rooting of both trees that minimizes the number of reticulations in the computed networks. While we do not suggest that a rooting chosen in this way will always be correct (and often there will be more

- *Daniel H. Huson is with the Center for Bioinformatics (ZBIT), University of Tübingen, Germany.*
  E-mail: daniel.huson@uni-tuebingen.de
- *Simone Linz is with the Department of Computer Science at the University of Auckland, New Zealand.*

Manuscript received ???; revised ???.

than one such rooting, in any case), the benefit here is that one may avoid obtaining a network that has a large number of reticulations that are only due to the improper rooting of one of the trees (as evident in row 1 of Table 1).

We first review some basic concepts in Section 2.1. In Section 2.2, we introduce the new concept of *autumn* trees and networks, which are phylogenetic trees and networks in which some of the taxa or leaves are alive and others are dead. We then define a number of operations that can be used to modify such trees and networks. These operations form the basis of our main algorithm for computing minimum hybridization networks, which is formulated in Section 2.3. We then provide some details on our implementation of the main algorithm and some variants, in Section 2.4.. In Section 3 we illustrate and discuss the straight-forward application of our approach to a number of examples, namely fish [8], grasses [19], Danthonioideae [33] and Nymphoides [20]. We finish with a brief discussion in Section 4.

In an Appendix, we prove the correctness of the new autumn algorithm by showing how the concepts defined in this paper are related to previous work done on *maximum-acyclic-agreement forest (MAAF)* for two rooted phylogenetic trees [9]–[14], [21].

2 METHODS

2.1 Phylogenetic Trees and Networks

In this section we introduce some basic concepts [22]. Throughout this paper, we will use $\mathcal{X}$ to denote a set of taxa. A *phylogenetic tree* T on $\mathcal{X}$ is a tree whose leaves are bijectively labeled by the elements of $\mathcal{X}$. Further, we call T *rooted*, if a node with indegree 0 and outdegree at least 2 in T has been designated as *root*. If T is rooted, then we can view T as a directed graph in which all edges are directed away from the root. An unrooted phylogenetic tree is usually assumed to have no nodes of degree two, while a rooted phylogenetic tree is usually assumed to have no node that has indegree one and outdegree one. In addition, a rooted phylogenetic tree is called *bifurcating* if all internal nodes have indegree one (or zero, for the root) and outdegree two.

Let T be a rooted phylogenetic tree on $\mathcal{X}$ and $e = \{v, w\}$ be an edge from v to w in T such that v lies on the path from the root to w. The set $C \subset \mathcal{X}$ of all taxa that label descendants of w is called the *cluster* associated with e or w. We use $\mathcal{C}(T)$ to denote the set of all clusters that are obtainable from T in this way. We say that a rooted phylogenetic tree T' on $\mathcal{X}$ *refines* T, or equivalently, that T' is a *refinement* of T, if every cluster of T is also a cluster of T', that is, $\mathcal{C}(T) \subseteq \mathcal{C}(T')$. In addition, T' is a called a *fully resolved* refinement if it is bifurcating. Note that any tree T is considered to be a refinement of itself.

Let T be a rooted phylogenetic tree on $\mathcal{X}$. If $\mathcal{X}'$ is a subset of $\mathcal{X}$, then we use $T(\mathcal{X}')$ to denote the minimal rooted subtree of T that connects all leaves that are labeled by taxa in $\mathcal{X}'$. Further, the subtree obtained from $T(\mathcal{X}')$ by contracting all non-root nodes of degree 2 is called the *restriction* of T to $\mathcal{X}'$ and is denoted by $T|_{\mathcal{X}'}$. We sometimes say that $T|_{\mathcal{X}'}$ is the subtree of T that is *induced* by $\mathcal{X}'$. A rooted phylogenetic tree S is said to be *pendant* in T, if S can be detached from a refinement of T by deleting a single edge (v, w), in which case we say that S is the *subtree* of T that is rooted at w.

Rooted phylogenetic networks are a generalization of rooted phylogenetic trees. One way to think of them is as a means of simultaneously representing a number of conflicting evolutionary histories. In more detail, a *rooted phylogenetic network* N on $\mathcal{X}$ is a rooted acyclic digraph with root ρ in which the leaves of N are bijectively labeled by the elements of $\mathcal{X}$. Any node that has indegree of at least 2 is called a *reticulate* node, whereas all others are called *tree* nodes. A *reticulate* edge is any edge that leads into a reticulate node, whereas all other edges are called *tree* edges.

Hybridization occurs in many animal and plant groups [23]. In instances where incomplete lineage sorting and phylogenetic error can be discounted, the incongruences between gene trees, which have been reconstructed for different parts of the genomes of the species under consideration, might be best explained by hybridization, which can result in introgression of genes and/or hybrid speciation [24], [25]. In this case, a phylogenetic network N on $\mathcal{X}$ that simultaneously explains all ancestral relationships of a set of gene trees, (i.e. rooted phylogenetic trees on subsets of $\mathcal{X}$) might be more appropriate to represent the evolutionary history of $\mathcal{X}$ than a rooted phylogenetic tree on $\mathcal{X}$. In the remainder of this paper, we will interchangeably refer to N as a *hybridization network* or a rooted phylogenetic network.

Let N be a hybridization network on $\mathcal{X}$, and let T be a rooted phylogenetic tree on $\mathcal{X}'$ such that $\mathcal{X}' \subseteq \mathcal{X}$. In the special case that N is *bicombining*, that is, that each reticulate node has indegree 2, we define the *hybridization number* of N as the number of reticulate nodes in N. In the more general case, in which nodes can have indegree greater than 2, the *hybridization number* of N is given by

$$h(N) = \sum_{v \neq \rho} (\text{indegree}(v) - 1),$$

where the sum is taken over all nodes v of N (except the root ρ) and $\text{indegree}(v)$ denotes the number of edges that lead into v. We say that a phylogenetic network N *displays* a phylogenetic tree T if there exists a rooted subtree of N that is a refinement of T. For two rooted phylogenetic trees T_1 and T_2 (both on arbitrary taxa sets), we set

$$h(T_1, T_2) = \min\{h(N) \mid N \text{ displays } T_1 \text{ and } T_2\}$$

and call $h(T_1, T_2)$ the *minimum hybridization number* of T_1 and T_2. Moreover, N is called a *minimum* hybridization network for T_1 and T_2 if it displays both trees and has exactly $h(T_1, T_2)$ reticulate nodes.

Now, let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. In this paper we consider the problem of computing a minimum hybridization network N on $\mathcal{X}_1 \cup \mathcal{X}_2$ that displays T_1 and T_2. As we only consider the case of two input trees, any such minimum network will be bicombining. Further, we not only aim at calculating a single minimum hybridization network that displays T_1 and T_2, but a *representative set* of such networks, as defined in [14]. In more detail, our new algorithm calculates one minimum hybridization network for each maximum-acyclic-agreement forest (for a formal definition, see the Appendix), that is, for each smallest-sized set of potential hybrid species, the algorithm returns one minimum hybridization network.

Given two rooted phylogenetic trees T_1 and T_2 on $\mathcal{X}$, the problem of determining $h(T_1, T_2)$ is called the HYBRIDIZATIONNUMBER problem, which is known to be NP-hard [9].

2.2 Autumn Trees and Networks

In this section we define the concept of autumn trees and networks as well as some operations that will play an important role in the remainder of this paper. In comparison to (ordinary) rooted phylogenetic trees and networks, each leaf of an autumn tree or network is either alive or dead which is helpful for the practical implementation of our algorithm.

An *autumn tree* $T := (T, A, D)$ consists of a phylogenetic tree T on $\mathcal{X}$ and an associated partitioning $A \cup D$ of the taxon set $\mathcal{X}$ into a set of *alive* taxa $A = A(T)$ and *dead* taxa $D = D(T)$. The leaves of T that are labeled by taxa in A are called *alive* whereas the other leaves, which are all labeled by elements of D, are called *dead*. Further below, we will allow any dead taxon to label more than one leaf. Any autumn tree can also be viewed as a (non-autumn) phylogenetic tree by simply ignoring the partitioning of leave into dead and alive.

An internal node v of T is called *alive*, if it is the ancestor of at least one alive leaf, and otherwise it is *dead*. It follows from this definition that v is alive, if and only if it has at least one alive child. Further, we say that v is an *alive branching node*, if v has at least two distinct children that are alive. A tree or subtree in which all nodes are alive, or dead, is called alive, or dead, respectively. Moreover, an edge leading to a dead node is called dead. For example, all three autumn trees shown in Figure 1 each have three alive branching nodes.

Let $T_1 := (T_1, A_1, D_1)$ and $T_2 := (T_2, A_2, D_2)$ be two autumn trees on $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. We will usually set $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ and assume $A_1 = A_2 \subseteq \mathcal{X}_1 \cap \mathcal{X}_2$, and will express this by saying that T_1 and T_2 are autumn trees on $\mathcal{X}$. We have both $A_1 \cup D_1 \subseteq \mathcal{X}$ and $A_2 \cup D_2 \subseteq \mathcal{X}$, although not necessarily equality in either case.

Recall that two rooted phylogenetic trees (autumn or not) are called *isomorphic*, if they have the same leaf-labeled topology. Let T_1 and T_2 be two autumn trees on $\mathcal{X}$ with alive taxa $A \subseteq \mathcal{X}$ and $A \neq \emptyset$. We say that T_1 and T_2 are *alive-isomorphic*, if the two rooted phylogenetic trees $T_1|_A$ and $T_2|_A$ induced by A are isomorphic. For example, all three autumn trees shown in Figure 1 are alive-isomorphic.

By extension, an *autumn network* N consists of a rooted phylogenetic network N on $\mathcal{X}$ and a partitioning $A \cup D$ of the taxon set $\mathcal{X}$ into a set $A = A(N)$ of alive taxa and set $D = D(N)$ of dead taxa, as above. The definitions of alive and dead nodes carry over from trees to networks.

While we require that each leaf of an autumn tree or network is labeled by exactly one taxon (which can be either dead or alive), we explicitly allow dead labels to appear more than once in the same tree or network.

In the following we describe a number of operations on autumn trees and networks that will be employed in our main algorithm.

2.2.1 Refinement

Let T_1 and T_2 be two autumn trees on $\mathcal{X}$. The *refinement* of T_1 and T_2 consists of two new autumn trees T_1' and T_2' that are defined as follows. We let T_1' be the autumn tree obtained from T_1 by inserting an appropriate additional edge for each alive cluster in T_2 that is compatible with all clusters in T_1, but is not already contained in T_1. The tree T_2' is obtained in a similar way.

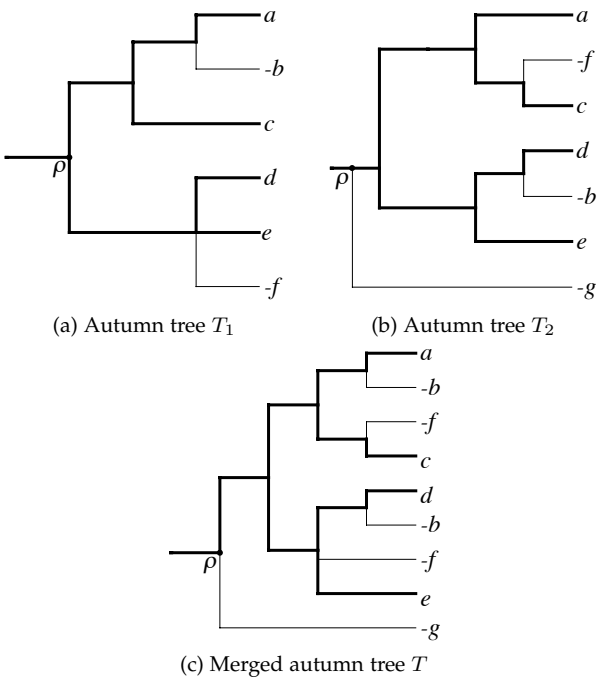


Fig. 1. **Merging of two autumn trees.** In (a) and (b) we show two autumn trees T_1 and T_2 on $\mathcal{X} = \{a, b, \ldots, g\}$, with each dead taxon indicated by a minus prefix and alive edges emphasized by bold lines. The two trees are alive-isomorphic and the result of merging them is the autumn tree T shown in (c). The root of each tree is labeled ρ.

2.2.2 Merge of Two Alive-Isomorphic Trees

Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}$. If they are isomorphic, then either of the two trees can be used to represent both trees (ignoring branch lengths, of course).

Now, assume that T_1 and T_2 are two autumn trees on $\mathcal{X}$ that are *alive-isomorphic*, but not isomorphic. In this case, one or both trees will contain dead leaves, that either occur only in one of the two trees, or that attach in different places in the two trees. For purposes of our algorithm, we require an autumn tree T that is alive-isomorphic to both T_1 and T_2, and that contains (refinements of) both trees. This tree is obtained by "merging" T_1 and T_2, and this operation is the main computational step in Algorithm 1.

In the following, we describe how to *merge* two alive-isomorphic autumn trees. Assume that T_1 and T_2 are two autumn trees on $\mathcal{X}$. If T_1 and T_2 are alive-isomorphic then there exists a bijection α between the alive branching nodes of both trees that preserves all ancestor-descendant relationships.

Consider two alive branching nodes v_1 and w_1 in T_1 such that v_1 is an ancestor of w_1. Choose v_1 and w_1 such that no other alive branching node lies on the path from v_1 to w_1 and consider the path $P_1 = (v_1 = x^1, x^2, \ldots, x^s = w_1)$ from v_1 to w_1, where $x^2, \ldots, x^{s-1}$ are all nodes on the path from v_1 to w_1, none of which an alive branching node. Note that each node x^i is the root of a dead subtree $T_1(x^i)$ that we obtain by ignoring the edge to x^{i+1}, for $i = 2, \ldots, s-1$.

Now consider $v_2 = \alpha(v_1)$ and $w_2 = \alpha(w_1)$ in T_2. By the properties of α, we have that v_2 and w_2 are alive branching nodes that are connected by a path $P_2 = (y^1 = v_2, y^2, \ldots, y^t = w_2)$ in T_2, in which $y^2, \ldots, y^{t-1}$ are all the nodes between v_2 and w_2, none of which are alive branching nodes. Again, each intermediate node y^i is associated with a dead subtree $T_2(y^i)$.

To merge two autumn trees T_1 and T_2 that are alive-isomorphic, there are three cases to be considered. In each case, we merge T_2 into T_1 to obtain the final tree, for ease of exposition:

1) The roots of the two trees are both alive branching nodes. In this case, we simply take each pair of nodes v_1 and w_1 as described above, together with their bijective counterparts v_2 and w_2, and merge the two associated paths P_1 and P_2 by inserting all the interior nodes $y^2, \ldots, y^{t-1}$ of P_2 into P_1, so as to preserve the order of all nodes $x^1 \ldots, x^s$ and $y^1 \ldots, y^t$. In this operation, the dead subtree $T_2(y^i)$ below each y^i remains attached to the node y^i. Also, any dead subtree rooted at v_2 or w_2 must be reattached under v_1 or w_1, respectively. Note that in general there will be many different possibilities for interleaving the members of P_1 and P_2.
2) The root of exactly one of the trees, say T_1, is not an alive branching node. Let w_1 be the first alive branching node below the root of T_1. Then the alive-isomorphism between T_1 and T_2 gives rise to a bijection between the subtree $T_1(w_1)$ rooted at w_1 and T_2, and we can use the construction indicated in (1) to merge T_2 into $T_1(w_1)$.
3) The root of neither tree is an alive branching node. In this case, the root ρ_1 of T_1 is connected to some alive branching node w_1 by a path $P = (x^1 = \rho_1, x^2, \ldots, x^s = w_1)$ with the property that none of the nodes $x^1, \ldots, x^{s-1}$ is an alive branching node. As above, each of these nodes is the root of a dead subtree $T_1(x^i)$, if we disregard the edge attaching x^i to x^{i+1}. Similarly, the root ρ_2 of T_2 is connected to some alive branching node w_2 by a path $P_2 = (y^1 = \rho_2, y^2, \ldots, y^t = w_2)$ with the same properties. To merge the two trees T_1 and T_2, we first merge the two paths P_1 and P_2 and then the two subtrees $T_1(w_1)$ and $T_2(w_2)$, in both cases as described in (1).

In Figure 1 we depict two autumn trees T_1 and T_2 on $\mathcal{X} = \{a, b \ldots, g\}$ that are alive-isomorphic, with dead taxa $D = \{b, f, g\}$. Case (2) applies to this example. The result of merging these two trees is the tree T.

Please note that the merge operation will produce an autumn tree in which a dead taxon will occur exactly twice, if it occurs in both input trees, and any alive taxon will occur exactly once. Moreover, we have the following property:

Lemma 1. *Let T_1 and T_2 be two autumn trees on $\mathcal{X}$. If T_1 and T_2 are alive-isomorphic and if T is the autumn tree obtained by merging T_1 and T_2 as described above, then T contains both T_1 and T_2 as subtrees.*

2.2.3 Subtree Reduction

In this section we describe the *subtree reduction* for autumn trees, which is a straight-forward extension of the subtree reduction for rooted phylogenetic trees described in [10]. The basic idea is simple. We search for two pendant subtrees of T_1 and T_2 that are alive-isomorphic and maximal such that both have at least two alive leaves. The two alive-isomorphic subtrees are then replaced by a single leaf.

Let T_1 and T_2 be two autumn trees $\mathcal{X}$ with alive taxa A. In a bottom-up fashion we consider each pair of alive branching nodes v_1 and v_2 contained in T_1 and T_2, respectively, with exception of the root nodes. There are three cases to be considered:

1) It may be that the subtrees $T_1(v_1)$ and $T_2(v_2)$ are alive-isomorphic.
2) More generally, it may be that one subtree, say $T_1(v_1)$, is alive-isomorphic to a *partial subtree* $T_2^*(v_2)$ rooted at v_2 that is obtained by using v_2 as the root and then attaching at least two alive children, together with their subtrees, below v_2. In other words, it may be that the subtrees rooted at v_1 and v_2 are alive-isomorphic, if we ignore some of the alive children of v_2.
3) In full generality, it may be that a partial subtree $T_1^*(v_1)$ is alive-isomorphic to a partial subtree $T_2^*(v_2)$. In other words, the subtrees rooted at v_1 and v_2 may be alive-isomorphic, if we ignore some of the alive children of v_1 and some of the alive children of v_2.

If the two nodes v_1 and v_2 fulfill one of these three conditions, but their parents do not, then we perform a subtree reduction, as described in the following for cases (1) and (3), while case (2) is a simple combination of the other two cases:

Case (1). Let Y be the set of alive taxa in $T_1(v_1)$. We delete all nodes and edges below v_1 (resp. v_2) in $T_1(v_1)$ (resp. $T_2(v_2)$) and label v_1 (resp. v_2) by a new formal taxon Y.

Case (3). Let Y be the set of alive taxa in $T_1^*(v_1)$. We delete all nodes and edges below v_1 (resp. v_2) in $T_1^*(v_1)$ (resp. $T_2^*(v_2)$), adjoin a new node w_1 (resp. w_2) to v_1 (resp. v_2) via a new edge, and then label w_1 (resp. w_2) by a new formal taxon Y.

This gives rise to two new trees T_1^+ and T_2^+ on $(\mathcal{X} \setminus Y) \cup \{Y\}$, which we call the *remainder* trees. In other words, we remove all taxa contained in Y from the set $\mathcal{X}$ and then add Y as a formal taxon to $\mathcal{X}$. Alternatively, in an implementation of this operation one could choose some fixed representative taxon $y \in Y$ to use as the label for v_1 and v_2.

As we want to be able to undo this reduction, we create two additional trees, T_1^- and T_2^-, which are obtained by hanging the two subtrees $T_1^*(v_1)$ and $T_2^*(v_2)$ below two new root nodes u_1 and u_2, respectively. We call these the *reduced* trees.

In summary, the subtree reduction takes as input two autumn trees T_1 and T_2 and, if applicable, produces as output four new trees T_1^+, T_2^+, T_1^- and T_2^- such that the latter two are alive-isomorphic, and the former two teaches each contain a leaf labeled by a formal taxon Y that specifies where the removed trees should be reattached. The formal taxon is not contained in any reducible subtree.

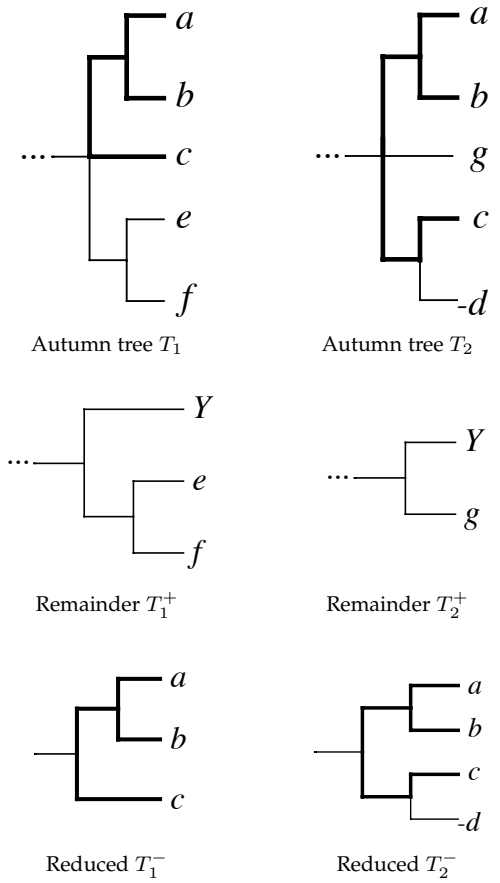


Fig. 2. **Subtree reduction of two autumn trees.** The two autumn trees T_1 and T_2 share a subtree on the set of alive taxa $Y = \{a, b, c\}$ and application of the subtree reduction gives rise to the two depicted remainder trees T_1^+ and T_2^+ and the two reduced trees T_1^- and T_2^-. All taxa are alive, except d, shown as -d. Edges contained in the shared subtree are shown in bold.

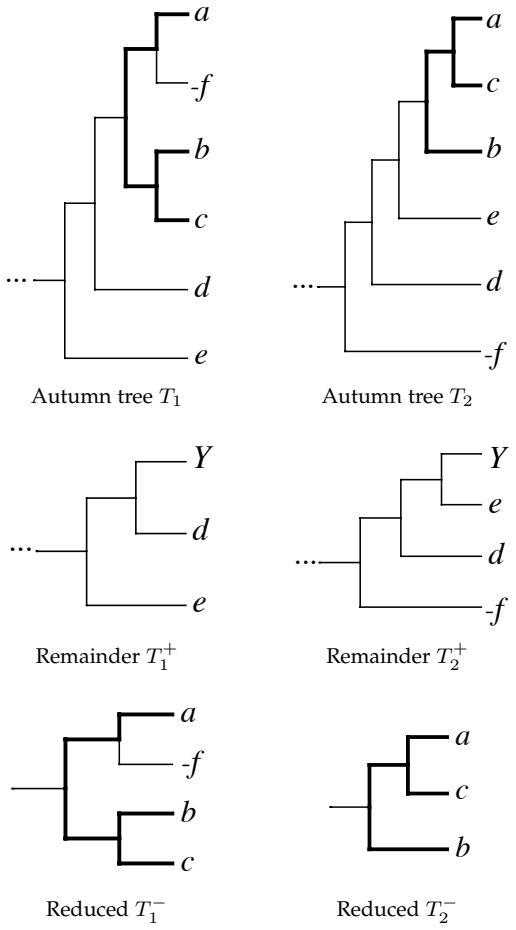


Fig. 3. **Cluster reduction of two autumn trees.** The two autumn trees T_1 and T_2 share the cluster $Y = \{a, b, c\}$ and application of the cluster reduction gives rise to the two depicted remainder trees T_1^+ and T_2^+ and the two reduced trees T_1^- and T_2^-. All taxa are alive, except f, shown as -f. Edges linking nodes labeled by taxa in Y are shown in bold.

An example of the subtree reduction is shown in Figure 2. The two autumn trees T_1 and T_2 share a common maximal subtree on the set of alive taxa $\{a, b, c\}$, as described in case (3) above. Subtree reduction gives rise to the depicted remainder and reduced trees. Here, the taxon Y plays the role of formal taxon in the two remainder trees.

2.2.4 Cluster Reduction

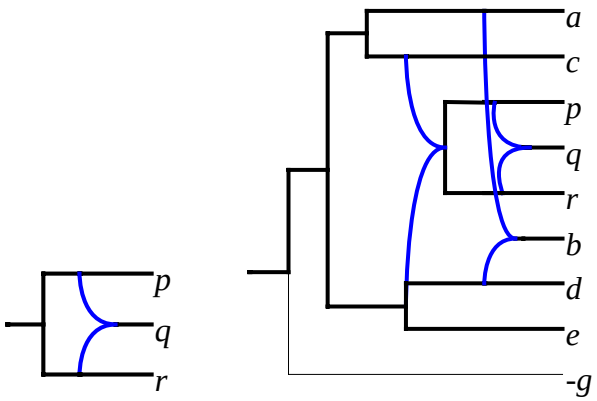
In this section we describe the *cluster reduction* for autumn trees, which is a straight-forward extension of the cluster reduction for rooted phylogenetic trees [26]. Roughly speaking, the cluster reduction is a weaker version of the subtree reduction that considers pendant subtrees that do not have a common binary refinement. Algorithmically, we search for a pair of minimal (partial) pendant subtrees that have the same set of at least two alive taxa. For such a pair of subtrees, we partition both trees by disconnecting the subtree from the rest of the tree.

Let T_1 and T_2 be two autumn trees on $\mathcal{X}$ with alive taxa A. Assume that all possible subtree reductions have already been performed so that the two trees are subtree irreducible. In a bottom-up fashion we consider each pair of alive branching nodes v_1 and v_2 contained in T_1 and T_2, respectively, with exception of the root nodes. There are three cases to consider.

1) It may be that the set of alive taxa below v_1 equals the set of alive taxa below v_2.
2) More generally, it may be that the set of alive taxa below v_1 equals the set of taxa of a partial subtree $T_2^*(v_2)$ rooted at v_2.
3) In full generality, it may be that a partial subtree $T_1^*(v_1)$ has the same set of alive taxa as a partial subtree $T_2^*(v_2)$.

In all three cases we can perform a cluster reduction. The actual construction is identical to that described above for a subtree reduction and gives rise to four trees, T_1^+, T_2^+, T_1^- and T_2^-. The former two trees each contain a leaf labeled by a formal taxon Y that corresponds to the set of alive taxa of the two latter trees. Further, the former two trees are called the *remainder* trees, while the latter two are called the *reduced* trees. As above, the formal taxon Y specifies where one should reattach the removed trees so as to obtain the original two trees.

An example of the cluster reduction is shown in Figure 3. The two autumn trees T_1 and T_2 share the minimal non-trivial cluster $\{a, b, c\}$ and the cluster reduction gives rise to the depicted remainder and reduced trees. Here, the taxon Y plays the role of formal taxon in the two remainder trees.



(a) Network N (b) Planted network N'

Fig. 4. **The planting operation.** By planting the taxon b and autumn network N (shown in (a)) into the autumn tree T (shown in Fig. 1(c)) at the nodes labeled $-b$ and $-f$, respectively, we obtain the new autumn network N' in (b).

2.2.5 Planting Taxa and Networks into Networks

Let N and N' be two autumn networks on $\mathcal{X}$ and $\mathcal{X}'$, respectively, with disjoint sets of alive taxa. Assume that N' contains a leaf, say v, that is labeled by an alive taxon Y that is a formal taxon and equals the set $A(N)$ of alive taxa in N. Then the operation of *planting* N into N' produces a new autumn network Q on $(\mathcal{X} \setminus \{Y\}) \cup Y$ that is obtained by identifying the root node of N with the node v in N' and removing the label Y from v.

Let N be an autumn tree or network on $\mathcal{X}$ and assume that x is a dead taxon that appears twice in N. Then the operation of *planting* the taxon x into N consists of identifying the two dead nodes that are labeled by x so as to obtain a new node v, attaching a new node w below v and then moving the label x to w. Moreover, the node x is declared to be alive.

As an example, consider the autumn tree T displayed in Figure 1(c) and assume that we would like to plant the taxon b in T, identifying the two nodes labeled $-b$. Moreover, assume that we want to plant the network N shown Figure 4(a) into T as well, identifying the two nodes labeled $-f$ with the root of N. The result is the autumn network N' that is shown in Figure 4(b).

2.2.6 Killing a Taxon

Let T be an autumn tree on $\mathcal{X}$. To *kill* an alive taxon $x \in \mathcal{X}$ simply means to move x from the set $A(T)$ of alive taxa to the set $D(T)$ of dead taxa. We use $T - \{x\}$ to denote the resulting tree.

2.3 The Autumn Algorithm

In this section we describe our main algorithm, which we call the *autumn algorithm* for hybridization networks, as it is based on autumn trees. It takes as input two rooted phylogenetic trees T_1 and T_2 on taxon sets $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. The trees are not required to be bifurcating. Moreover, we do not require the two taxon sets to be identical, however they must have at least one taxon in common, and should ideally share many taxa, if the calculation is to make sense. The output of the algorithm is a representative set of all minimum hybridization networks that contain T_1 and T_2, as defined in Section 2.4.1. The algorithm is based on the operations described above.

In a preprocessing step, we transform T_1 and T_2 into autumn trees by declaring all taxa that appear in both trees as alive and all others as dead, that is, we set $A(T_1) = A(T_2) = \mathcal{X}_1 \cap \mathcal{X}_2$, $D(T_1) = \mathcal{X}_1 \setminus \mathcal{X}_2$ and $D(T_2) = \mathcal{X}_2 \setminus \mathcal{X}_1$. We then refine both trees; i.e. we use T_2 as a guide to refine T_1 as much as possible and vice versa. The algorithm described below will return a list of autumn networks $\mathcal{N} = \{N_1, \ldots, N_k\}$ on $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$ such that $A(N_i) = \mathcal{X}_1 \cap \mathcal{X}_2$ and $D(N_i) = D(T_1) \cup D(T_2)$ holds for all networks in the output. Moreover, each dead taxon will appear only once as a label in any of the networks. By simply declaring all dead taxa to be alive, we obtain a set of networks that contain all taxa and display both trees T_1 and T_2.

After preprocessing, the algorithm operates recursively. At the beginning of the recursion, we are given two autumn trees T_1 and T_2, a set of possible hybrid taxa H (that is, alive taxa that correspond to possible hybrid leaves) and a non-negative integer k, which specifies the maximum number of hybridization events that we are allowed to use. Here is an overview of the three steps to be performed in each recursion:

1) We first check whether the two trees are subtree reducible. If they are, we apply the subtree reduction and then recursively call the algorithm on the two remainder trees. We merge the two reduced trees, plant the merged tree into the networks obtained by the recursion, and then return the resulting networks.
2) We then check whether the two trees are cluster reducible. If they are, we obtain two sub-problems that are both processed recursively. We then plant all networks returned for the pair of reduced trees into the networks returned for the two remainder trees and return the set of resulting networks.
3) If neither of the two above steps apply, then for each taxon x in H, we recursively run the algorithm on the refinement of $T_1 - \{x\}$ and $T_2 - \{x\}$, the two trees obtained by killing x in the two input trees, and $H' = H \setminus \{x\}$. We return all networks obtained in this way that have a minimum number of reticulations and for which this number is less or equal to the given bound k, after planting x into each of them.

The algorithm is originally called on two autumn trees T_1 and T_2 that have the same set of alive taxa, after initial preprocessing, and have been refined. We will use H to denote the set of potential hybrid taxa, that is the set of alive taxa that correspond to potential hybrid leaves. An upper bound k for the number of reticulate nodes to be used is also required. (E.g. such an upper bound is the number of reticulate nodes contained in the *cluster network* computed for the set of clusters contained in T_1 and T_2 (see [27]). Here is a more detailed description of the autumn algorithm in pseudo-code:

Algorithm 1 (AUTUMN(T_1, T_2, H, k))**.**

Input: *Autumn trees T_1 and T_2 on $\mathcal{X}$, set of possible hybrids $H \subset \mathcal{X}$ and a maximum number k of hybridizations to be used.*
Output: *Set of autumn networks $\mathcal{N}$ on $\mathcal{X}$ and the number h of hybridizations contained in each network in $\mathcal{N}$, with $h \leq k$.*
if *T_1 and T_2 are subtree reducible* **then**
  Let T_1^+, T_2^+, T_1^- and T_2^- be the four resulting trees
  and let y be the formal taxon placed in T_1^+ and T_2^+
  Recursively call AUTUMN$(T_1^+, T_2^+, H \cup \{y\}, k)$ *to obtain $\mathcal{N}$ and h*
  Let T be the tree obtained by merging T_1^- and T_2^-
  if *$h \leq k$* **then**
    for each *network $N \in \mathcal{N}$* **do**
      Plant T into N
    return *the resulting set of autumn networks and the number of hybridizations h*
  else
    return *$\emptyset$*
if *T_1 and T_2 are cluster reducible* **then**
  Let T_1^+, T_2^+, T_1^- and T_2^- be the four resulting trees
  and let y be the formal taxon placed in T_1^+ and T_2^+
  Recursively call AUTUMN$(T_1^+, T_2^+, H \cup \{y\}, k)$ *to obtain $\mathcal{N}^+$ and h^+*
  Recursively call AUTUMN$(T_1^-, T_2^-, H, k - h^+)$ *to obtain $\mathcal{N}^-$ and h^-*
  if *$h^+ + h^- \leq k$* **then**
    for each *autumn network $N^+ \in \mathcal{N}^+$* **do**
      for each *autumn network $N^- \in \mathcal{N}^-$* **do**
        Plant N^- into N^+
    return *the set of resulting autumn networks and $h = h^+ + h^-$*
  else
    return *$\emptyset$*
Set $\mathcal{N} = \emptyset$
for each *taxon $x \in H$* **do**
  Let T_1' and T_2' be the two trees obtained by refining $T_1 - \{x\}$ and $T_2 - \{x\}$
  Recursively call AUTUMN$(T_1', T_2', H \setminus \{x\}, k-1)$ *to obtain $\mathcal{N}'$ and h*
  if *$h < k$* **then**
    Set $\mathcal{N} = \emptyset$ and set $k = h$
  if *$h \leq k$* **then**
    Plant the taxon x into each network in $\mathcal{N}'$ and add the results to $\mathcal{N}$
return *$\mathcal{N}$ and k*

2.4 Implementation Details

We have implemented the algorithm within the framework of the Java program Dendroscope 3 [28]. More precisely, we have produced two different versions of the algorithm. The first version aims only at computing the minimum hybridization number for two rooted phylogenetic trees and we will refer to this as the *autumn-number* calculation. The algorithm is obtained by simplifying Algorithm 1 accordingly. The second version, *autumn-networks*, implements the full algorithm. Unsurprisingly, autumn-number runs substantially faster than autumn-networks because it does not have to keep track of computed networks and only needs to enter a recursion if there is hope that it will produce a strictly better hybridization number than already computed.

The autumn-number version is implemented in parallel. In each cluster reduction, both sub-problems are run in different threads. Threads communicate with each other so that each sub-problem takes into account how many reticulate nodes its complementary sub-problems requires. Similarly, different taxa are killed in parallel and competing threads are aware of the best value attained by their competitors.

To provide autumn-number with an initial value (upper bound) for k, the number of reticulations that may be used, we compute the cluster network [27] for the set of clusters in the two input trees and set k to be equal to the number of reticulate nodes in that network. To provide an initial value for k in autumn-networks, we first run the autumn-number algorithm and then set k to the number computed by autumn-number.

To speed up the recursive call in both versions we cache all results computed on any sub-problem and before calling the algorithm recursively, we first determine whether the sub-problem is already present in the cache. We use a cache of bounded size and remove the least recently used element whenever necessary to make room for new entries.

Given two unrooted phylogenetic trees, we provide an implementation of the algorithm that allows one to determine a rooting of both trees so as to minimize the number of hybridizations [29]. To address this problem, we determine all possible pairs of rootings of both trees and then order these by decreasing level of midpoint-rootedness (if no branch lengths are given, we scale each branch to have branch length 1). We run the autumn-number algorithm, using the cache across different runs (for different rootings). As many of the sub-problems are shared across different rootings, this results in a significant speed up of the calculation. Unsurprisingly, the algorithm can not always be run to completion because some of the possible choices may correspond to very high minimum hybridization numbers, as illustrated below.

As illustrated in the next section, the autumn algorithm can produce hundreds of optimal hybridization networks for a given pair of input trees. How to determine which of them are biologically relevant? Often, when studying closely related species, hybridization is speculated based on observations of morphology, sympatry of species, or knowledge of their reproductive compatibility. To make use of such additional biological information, we have implemented a new command

```
rankNetworks [mode=<rank|filter>]
[noHybrid=<name,...>] [noRecentHybrid=<name,...>]
[hybrid=<name,...>] [recentHybrid=<name,...>]
```

in Dendroscope 3 [28] that allows one to specify four sets of taxa, namely:

1) those taxa that should not occur below any reticulate node,
2) those taxa that should not occur as the direct descendant of any reticulate node,
3) those taxa that should occur below some reticulate node, but not necessarily as a direct descendant, and
4) those taxa that should occur as the direct descendant of some reticulate node.

When specifying at least one of these sets, the user can ask the program either to *rank* all networks by the number of taxa for which the specified sets are consistent with a network, or to *filter* the networks so as to obtain only those networks for which all specified sets are consistent.

2.4.1 MAAF Filtering

Our aim in this paper is to produce a *representative* set of minimum hybridization networks for a pair of input trees, as defined in [14]. In more detail, such a set is given by providing exactly one network for each MAAF (*maximum-acyclic-agreement forest*, see the Appendix for details) associated with the two given input trees. The autumn algorithm computes at least one hybridization network per MAAF, but often more than one. To ensure that only one network is reported per MAAF, our implementation compares the MAAFs associated with the computed networks and removes all redundant networks.

3 RESULTS

In this section we illustrate how Algorithm 1 can easily be used to explore possible hybridization scenarios given two conflicting gene trees, using our implementation of the autumn algorithm in Dendroscope 3. The software and datasets used in this paper can be downloaded here: http://ab.inf.uni-tuebingen.de/data/supp/autumn .

3.1 Hybridization in Lamprologine Cichlids

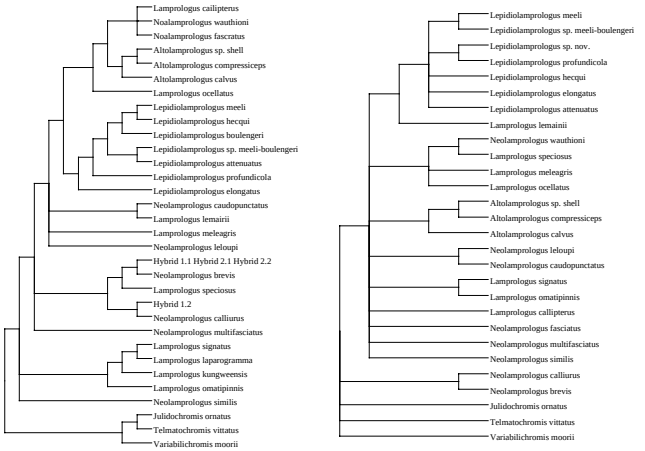
In [8], the authors study the question of whether the species within a certain monophyletic lineage of lamprologine cichlids (Lake Tanganyikas shell dwellers) might be particularly prone to accidental hybridization, due to the fact that their distinctive life style enables them to live and breed in closest vicinity. To address this, among other arguments, the authors present a strict consensus tree of an analysis based on sequence data set of the mitochondrial gene *ND2* for 48 taxa and a rooted phylogenetic tree based on an AFLP data set for 47 taxa, displayed in Figure 2 of [8]. Based on a comparison of the two trees, they deduce four putative hybrid species, namely *L. meleagris*, *N. wauthioni*, *L. speciosus*, and *N. multifasciatus*. The authors also provide a hybridization network that they produced by hand, displayed in Figure 2 of [8].

To reanalyze this data using our algorithm, we proceeded as follows. We entered the two trees displayed in Figure 2 of [8] into Dendroscope 3 by hand and contracted any edges that have a bootstrap support of 50% or less. Because both trees are *multi-labeled*, that is, contain the same taxon more than once (corresponding to different individuals of the same taxonomic group), the next step was to convert the two trees into single-labeled trees. For this we used a feature of Dendroscope 3 that takes as input a multi-labeled tree and produces as output a single-labeled cluster network [22], [27]. In this case, for both trees, the resulting cluster network is a rooted phylogenetic tree. The rooting of the two trees differs slightly, and so the next step was to make a minor adjustment to the rooting of the second tree so as to match the first tree. The two resulting trees are shown in Figure 5. Note that both trees are multifurcating and have taxon sets that differ by 14 taxa.

Application of the autumn algorithm to the two trees gives rise to a representative set of twelve hybridization networks, each containing four reticulate nodes. The calculation took less that five seconds. The resulting list of networks is shown in Figure 6.

Our analysis confirms that the number of hybridization events required to reconcile the two trees is indeed four. However, the network displayed in Figure 2 of [8] does not appear in the list of twelve networks that our algorithm produced. The reason for this may be that perhaps the authors of the original paper aimed at a scenario in which the hybridization events happen between recent lineages and thus allowed some inaccuracies in the containment of the two trees to achieve their goal. All twelve optimal hybridization networks computed by the autumn algorithm contain at least one hybridization event involving an ancient lineage.

To determine which of the computed networks is most similar to the one presented in [8], we applied the



(a) Mitochondrial tree (b) Nuclear tree

Fig. 5. **Two input trees.** Two rooted phylogenetic tree based on (a) the mitochondrial gene *ND2* and (b) AFLP data from the nuclear genome, for cichlids, after preprocessing as described in the text (based on Figure 2 of [8]).

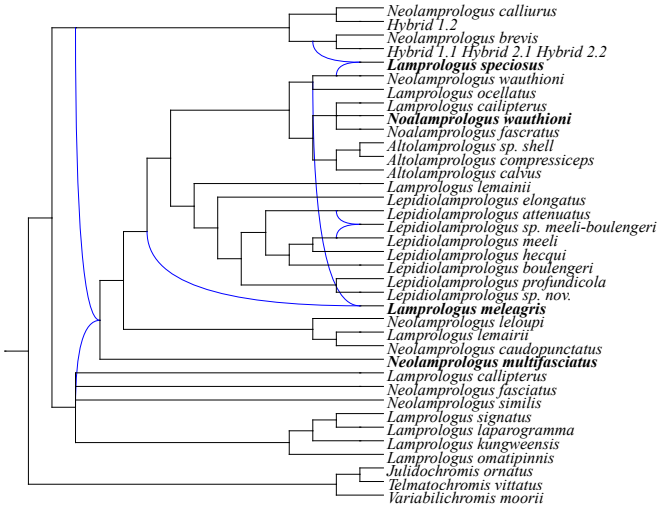


Fig. 7. **Top hybridization network.** Using our `orderNetwork` feature in Dendroscope, we rank all 12 networks shown in Figure 6 by how many of the four putative hybrid species (highlighted in bold) appear as "recent hybrids", or at least below some hybrid node. While all four do appear below a (common) hybrid node, only two of the four are displayed as recent hybrids.

`rankNetworks` command described in Section 2.4 to rank the networks by how many of the four putative hybrid species (mentioned above) occur below reticulate nodes. The first network in the list (shown in Figure 7) displays two of the four putative hybrid species as "recent hybrids" and all four are placed below a common hybridization node. In contrast to the results reported in [8], *L. sp. meeli-boulengeri* appears as a hybrid.

3.2 Grass Data Sets

A publication on the "phylogeny and subfamilial classification of the grasses (Poaceae)" [19] uses six different gene trees to classify different grasses, based on the following genes: *ITS*, *ndhF*, *phyB*, *rbcL*, *rpoC2* and *GBSSI*. The trees

Fig. 6. **Minimum hybridization networks.** All twelve minimum hybridization networks for the two trees shown in Figure 5.

displayed in that paper are nearly all multifurcating and contain overlapping, but different, taxon sets.

A number of papers on methods for computing hybridization networks have used simplified versions of these trees as test sets [11], [14], [30]–[32]. The modified trees are bifurcating, and are grouped into pairs of induced trees on the same taxon sets. Moreover, probably by mistake, the rooting of some of the trees has been changed compared to the original publication. We show the minimum hybridization number and size of a representative set of hybridization networks for each such pair of simplified gene trees in column (b) of the table shown in Table 1.

Because in some cases the rooting of the pairs of test trees differs significantly from the original trees, we ran our rerooting algorithm on each pair of trees so as to find the rooting that minimizes the number of reticulations and show the results in column (c) in Figure 1. In most cases, the minimum coincides with the number that one obtains when using the original rooting of the two trees.

With the new autumn algorithm presented in this paper, none of the above data simplifications is necessary and one can easily compare each pair of trees as originally

(a) Pair of genes	(b) Bifurcating equal taxa h	(b) Bifurcating equal taxa N	(c) Optimal rooting h	(d) Original multifurcating h	(d) Original multifurcating N
ndhF phyB	14	2268	7	6	54
ndhF rbcL	13	48	8	7	36
ndhF rpoC2	12	27	9	5	108
ndhF GBSSI	9	396	6	14	12
ndhF ITS	19	81	≤ 17	4	135
phyB rbcL	4	4	4	4	4
phyB rpoC2	7	1	4	3	18
phyB GBSSI	3	6	2	3	6
phyB ITS	8	9	8	8	15
rbcL rpoC2	13	9	7	5	108
rbcL GBSSI	7	35	3	3	12
rbcL ITS	14	156	≤ 11	8	18
rpoC2 GBSSI	1	1	1	2	2
rpoC2 ITS	15	246	≤ 12	10	36
GBSSI ITS	8	18	5	7	259

Table 1. For each pair of genes in column (a), we report the minimum hybridization number h and number of networks N for trees based on [19]. In more detail, in column (b), we show results for the simplified trees obtained from the original trees by removing any taxa not shared by both gene trees and then adding branches so as to make the trees binary, as used in a number of studies including [11], [14], [30]–[32]. In (c), we report the hybridization number for the same trees rerooted so as to minimize the required number of reticulations. Finally, in (d), we report the results obtained from the original trees reported in [19], containing different numbers of taxa and multifurcations. In three cases, the computation of the optimal rooting did not complete within a couple of days and was then aborted, in these cases the number listed in column (c) is only an upper bound.

presented in [19]. To illustrate this, we first captured all six gene trees using Dendroscope 3, as the original tree files are no longer available. For each pair of trees, we ran the autumn algorithm so as to obtain a representative set of all hybridization networks for the pair of trees. The results are reported in column (d) in Table 1.

In nearly all cases, using the original, multifurcating trees produces hybridization networks with lower hybridization numbers than using the "simplified" bifurcating ones. This is to be expected because it is seldom the case that all edge in a fully resolved tree can be inferred with confidence, and incorrectly inferred edges cause additional reticulations.

In a more recent paper [33], the authors explore the potential impact of conflicting gene trees on inferences of evolutionary history above the species level, in particular studying grasses and using both chloroplast DNA (cpDNA) and nuclear ribosomal DNA (ITS region). The authors argue that the contradictions can be explained by past hybridization events, which have linked gains of complex morphologies with unrelated chloroplast lineages and have erased evidence of dispersals from the nuclear genome. To go beyond the analyses performed in [33], using the autumn algorithm one can compute a representative set of all minimum hybridization network for the two trees. In 12 seconds our implementation establishes that the minimum number of reticulations is 13 and there are 729 such networks.

As an another example, the authors of [20] investigate the phylogenetic relationships in Nymphoides (a genus of aquatic flowering plants) and report two multifurcating gene trees, one based on the ITS region and a second based on the matK/tmK gene. Application of the autumn algorithm to these two trees results in a hybrid number of 11 and 1080 different minimum hybridization networks (computation took 8 seconds).

We end this section by noting that Table 1 contains three gene tree pairs (ndhf rbcL, rbcL rpoC2 ndhF rpoC2) for which both trees were reconstructed from chloroplast DNA and one does consequently not expect any gene tree incongruence that is due to hybridization.

4 Discussion

The use of rooted phylogenetic networks in systematics has been hampered by the lack of appropriate tools for their computation. Existing algorithms usually require that the input trees are fully resolved and/or have identical taxon sets. Moreover, they are often only implemented as command-line tools that lack interactive features.

Our algorithm, as implemented in Dendroscope 3, overcomes previous practical shortcomings and can be applied to realistic phylogenetic trees. It helps to provide a unique interactive platform for researchers to explore the application of rooted phylogenetic networks to their phylogenetic trees and data.

The concept of an autumn tree, as introduced in this paper, provides a new way of formulating algorithms on rooted phylogenetic trees and networks, as shown in this paper. Our algorithm for computing the hybridization number is a parallel algorithm that can take advantage of current multicore computer architectures.

Future work in this area will address the question of extending the results to an input of more than two trees, and will also explore the problem of allowing multiple occurrences of the same taxon. Moreover, the question of how best to root the network, in the case that the rooting of the input trees is unknown, requires further investigation.

The number of hybridizations produced is usually too large to be easily inspected by eye. While the filter and ranking procedure described above and implemented in Dendroscope 3 can help biologists to focus on those networks that fulfill constraints based on known hybrids, more work needs to be done to help reduce the number of networks that require inspection.

Appendix
Correctness of the Algorithm

Previous work on the computation of hybridization networks is formulated in terms of *maximum-acyclic-agreement forests* (MAAFs) [9]–[14], [21] . While MAAFs are a powerful tool for developing mathematical insights, they are perhaps less suited for developing a practical algorithm for solving the problem, because they discard information on where reticulate nodes attach. Autumn trees, as introduced in this paper, explicitly maintain this information (namely as dead taxa) and so we believe that they may provide a more suitable data structure for formulating our main result, Algorithm 1. The goal of this section is to prove the correctness of Algorithm 1. This is done by showing how the algorithm can be interpreted in terms of the calculation of MAAFs and referring to appropriate results in the literature.

We explicitly prove that our algorithm can also handle non-bifurcating trees.

We start with some additional definitions. A characterization of the HYBRIDIZATIONNUMBER problem for two rooted phylogenetic trees on the same set of taxa in terms of so-called *agreement forests* was first introduced in [21] for rooted bifurcating phylogenetic trees and was then later generalized to arbitrary rooted phylogenetic trees [10]. Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}$. The set $\mathcal{X}$ is often referred to as the *label set* of T_1 and T_2 and denoted by $\mathcal{L}(T_1) = \mathcal{L}(T_2)$. For the purposes of the upcoming definitions, we regard the root of both T_1 and T_2 as a node labeled ρ at the end of a pendant edge adjoined to the original root. Further, we also regard ρ as part of the label set of T_1 and T_2, thus we view their label sets as $\mathcal{X} \cup \{\rho\}$.

A *forest* for T_1 is a partition $\{\mathcal{L}_\rho, \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$ of its label set $\mathcal{X} \cup \{\rho\}$, where $\mathcal{L}_\rho$ contains ρ, no part is empty, and the trees in $\{T_1(\mathcal{L}_i) : i \in \{\rho, 1, 2, \ldots, k\}\}$ are edge-disjoint rooted subtrees of T_1. An *agreement forest* $\mathcal{F}$ for T_1 and T_2 is a forest $\{\mathcal{L}_\rho, \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$ for T_1 and T_2 such that, for all $i \in \{\rho, 1, 2, \ldots, k\}$, the trees $T_1|_{\mathcal{L}_i}$ and $T_2|_{\mathcal{L}_i}$ have a common bifurcating refinement. To exclude the possibility that species inherit genetic material from their own descendants we need an additional constraint. Let $\mathcal{F} = \{\mathcal{L}_\rho, \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$ be an agreement forest for two arbitrary rooted phylogenetic trees T_1 and T_2 on $\mathcal{X}$. Further, let $G_\mathcal{F}$ be the directed graph that has node set $\mathcal{F}$ and an arc $(\mathcal{L}_i, \mathcal{L}_j)$ precisely if $i \neq j$ and either

(I) the path from the root of $T_1(\mathcal{L}_i)$ to the root of $T_1(\mathcal{L}_j)$ contains an edge of $T_1(\mathcal{L}_i)$, or
(II) the path from the root of $T_2(\mathcal{L}_i)$ to the root of $T_2(\mathcal{L}_j)$ contains an edge of $T_2(\mathcal{L}_i)$.

We say that $\mathcal{F}$ is an *acyclic-agreement forest* for T_1 and T_2 if $G_\mathcal{F}$ contains no directed cycle and we call $G_\mathcal{F}$ the *inheritance graph* of $\mathcal{F}$ with regards to T_1 and T_2. If $\mathcal{F}$ contains the smallest number of parts over all acyclic-agreement forests for T_1 and T_2, we say that $\mathcal{F}$ is a *maximum-acyclic-agreement forest* for T_1 and T_2, and we denote this number by $m_a(T_1, T_2)$. The proof of the next theorem is given in Theorem 4.1 of [10]

Theorem 1. *Let T and T' be two rooted phylogenetic trees on $\mathcal{X}$. Then*

$$h(T, T') = m_a(T, T').$$

This characterization is crucial to many of the computational results associated with solving instances of HYBRIDIZATIONNUMBER that are cited above.

To simplify notation, in the following we use $T[-x]$ to denote the rooted phylogenetic tree $T|_{\mathcal{X}\setminus\{x\}}$ obtained from a given rooted phylogenetic tree T on $\mathcal{X}$ by removing the leaf labeled x from T. Our first aim is to show why the subtree and cluster reductions preserve the minimum hybridization number of two rooted phylogenetic trees.

Lemma 2. *Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}$, and let T_1^+ and T_2^+, T_1^- and T_2^- be the two tree pairs that result from applying a subtree or cluster reduction to T_1 and T_2. Then*

$$h(T_1, T_2) = h(T_1^+, T_2^+) + h(T_1^-, T_2^-).$$

Moreover, in the case of a subtree reduction we have $h(T_1^-, T_2^-) = 0$.

The correctness of this result follows from Proposition 6.1 of [10] and Proposition 5.16 of [26].

In [11], the authors describe an exact algorithm that calculates the minimum hybridization number for two bifurcating rooted phylogenetic trees. Essentially, they show that if each recursion R of their algorithm is applied to a pair of subtree-reduced rooted bifurcating phylogenetic trees T_1 and T_2 on $\mathcal{X}$, then the number of possibilities that is needed to be checked in R can be reduced from $2|\mathcal{X}| - 1$ (which naively considers each edge of either T_1 or T_2) to $|\mathcal{X}|$ because there exists an $x \in \mathcal{X}$ such that the minimum hybridization number for T_1 and T_2 equals that number for $T_1[-x]$ and $T_2[-x]$ plus 1. Intuitively, R branches into $|\mathcal{X}|$ computational paths, each considering a distinct leaf of T_1 and T_2. The next two lemmas show that this result can be generalized to a pair of arbitrary rooted phylogenetic trees on $\mathcal{X}$.

Lemma 3. *Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}$. Suppose that T_1 and T_2 have no common pendant subtree whose leaf set size is at least 2. Then, for each $x \in \mathcal{X}$, we have*

$$h(T_1, T_2) \leq h(T_1[-x], T_2[-x]) + 1.$$

Proof. Let $\mathcal{F}_x$ be a maximum-acyclic-agreement forest for $T_1[-x]$ and $T_2[-x]$. Clearly,

$$\mathcal{F} = \mathcal{F}_x \cup \{x\}$$

is an acyclic-agreement forest for T_1 and T_2. Moreover, we have $|\mathcal{F}| = |\mathcal{F}_x| + 1$. Thus

$$h(T_1[-x], T_2[-x]) + 1 = |\mathcal{F}_x| - 1 + 1 = |\mathcal{F}| - 1 \geq h(T_1, T_2).$$

This establishes the lemma. □

Lemma 4. *Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}$. Suppose that T_1 and T_2 have no common pendant subtree whose leaf set size is at least 2. Then there exists a taxon $x \in \mathcal{X}$ such that*

$$h(T_1, T_2) = h(T_1[-x], T_2[-x]) + 1.$$

Proof. Let $\mathcal{F} = \{\mathcal{L}_\rho, \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_k\}$ be a maximum-acyclic-agreement forest for T_1 and T_2. First observe that the the inheritance graph $G_\mathcal{F}$ of $\mathcal{F}$ with respect to T_1 and T_2 have a node $\mathcal{L}_i$ with $i \in \{\rho, 1, 2, \ldots, k\}$ whose out-degree is zero because the graph is acyclic. Further, since T_1 and T_2 have no common pendant subtree whose leaf set size is at least 2, $\mathcal{L}_i$ is a singleton in $\mathcal{F}$. As ρ is never a singleton in $\mathcal{F}$ by Lemma 1 of [21], we may assume that $\mathcal{L}_i = \{x\}$, where $x \in \mathcal{X}$.

Since $\mathcal{F}$ is an acyclic-agreement forest for T_1 and T_2, it follows that

$$\mathcal{F}_x = \mathcal{F} - \{x\}$$

is an acyclic-agreement forest for $T_1[-x]$ and $T_2[-x]$. Further, we have $|\mathcal{F}| = |\mathcal{F}_x| + 1$ and this implies

$$h(T_1, T_2) = |\mathcal{F}| - 1 = |\mathcal{F}_x| + 1 - 1 \geq h(T_1[-x], T_2[-x]) + 1. \quad (1)$$

Combining (1) with Lemma 3 completes the proof. □

In view of the proof of Lemma 4, the calculation of the minimum hybridization number $h(T_1, T_2)$ for two rooted phylogenetic trees T_1 and T_2 on $\mathcal{X}$ can be simply described as the repeated application of the following steps: First,

apply all possible subtree reductions to T_1 and T_2 so as to obtain two reduced trees T_1^+ and T_2^+. Second, for each taxon x contained in T_1^+ and T_2^+, delete x from both trees so as to obtain two new trees T_1 and T_2. If T_1 and T_2 are isomorphic then stop; otherwise recurse on T_1 and T_2. The minimum hybridization number of the original two input trees is given by the minimum depth of recursions that is needed to reach a pair of isomorphic trees.

For any execution of this algorithm that stops at the minimum recursion depth, the corresponding maximum-acyclic-agreement forest can be obtained as the union of the label set of the tree T_1 (in the last recursion) and the set of all labels x that have been deleted at some level of the recursion, after undoing all subtree and cluster reductions.

Rather than explicitly removing taxa, Algorithm 1 uses autumn trees and the operation of killing taxa. Accordingly, the goal of minimizing the number of deleted taxa is replaced by the goal of minimizing the number of killed taxa. Moreover, the test of whether T_1 and T_2 are isomorphic is replaced by the test of whether they are alive-isomorphic.

It follows from the structure of Algorithm 1 and Lemma 2 (for a cluster reduction) that the output of the algorithm is a set $\mathcal{N}$ of hybridization networks whose number of hybridization nodes is $h(T_1, T_2)$, where T_1 and T_2 are the two original input trees. Because the operation of merging two alive-isomorphic autumn trees preserves dead leaves and branches, it follows that any hybridization network in $\mathcal{N}$ will display T_1 and T_2.

Until now, we have restricted our attention to the calculation of minimum hybridization networks for two rooted phylogenetic trees on $\mathcal{X}$. However, many biological data sets contain phylogenetic trees whose taxa sets are not identical but only overlap on a subset of taxa. We will now discuss why the autumn algorithm can also deal with such data. Let T_1 and T_2 be two rooted phylogenetic trees on $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. If $\mathcal{X}_1 \neq \mathcal{X}_2$, then in a preprocessing step we kill all those taxa that are exclusively contained in $\mathcal{X}_1$ or $\mathcal{X}_2$. The algorithm then calculates minimum hybridization networks for the two resulting autumn trees on $\mathcal{X}_1 \cap \mathcal{X}_2$. Finally, for each resulting network, the algorithm turns each dead taxon into an alive taxon. Clearly, the resulting networks display T_1 and T_2. Further, the next theorem shows that each of these networks is indeed a minimum hybridization network for T_1 and T_2.

Theorem 2. *Let T_1 and T_2 be two rooted phylogenetic tree on $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively. Then,*

$$h(T_1, T_2) = h(T_1|_{(\mathcal{X}_1 \cap \mathcal{X}_2)}, T_2|_{(\mathcal{X}_1 \cap \mathcal{X}_2)}).$$

Before proving Theorem 2, we need some additional definitions. Let T be a rooted phylogenetic tree on $\mathcal{X}$, and let $\mathcal{Y} \subseteq \mathcal{X}$. Further, let $\mathcal{P} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ be a partition of $\mathcal{X} \setminus \mathcal{Y}$ such that, for each $i \in \{1, 2, \ldots, n\}$, $T|\mathcal{X}_i$ is a maximal pendant subtree of T that exclusively contains leaves that are labeled with elements in $\mathcal{X} \setminus \mathcal{Y}$. We call $\mathcal{P}$ the *pendant partition* of T with regards to $\mathcal{X} \setminus \mathcal{Y}$. Note that $\mathcal{P}$ is well-defined.

Proof of Theorem 2: Throughout this proof, let $\mathcal{Y} = \mathcal{X}_1 \cap \mathcal{X}_2$. Assume that $\mathcal{Y} = \emptyset$. Let N be the hybridization network obtained from T_1 and T_2 by joining the root of T_1 and the root of T_2 by a new edge e, subdividing e with a new node v, and regarding v as the root of N. Noting that $h(N) = 0$ and that N displays T_1 and T_2, the result clearly follows.

Therefore, we may assume for the rest of this proof that $\mathcal{Y} \neq \emptyset$. We first show that

$$h(T_1, T_2) \geq h(T_1|_{\mathcal{Y}}, T_2|_{\mathcal{Y}}). \quad (2)$$

Let N be a hybridization network on $\mathcal{X}_1 \cup \mathcal{X}_2$ that displays T_1 and T_2 and has exactly $h(T_1, T_2)$ reticulate nodes. Since, N also displays $T_1|_{\mathcal{Y}}$ and $T_2|_{\mathcal{Y}}$, Equation 2 immediately follows.

Second, we show that

$$h(T_1, T_2) \leq h(T_1|_{\mathcal{Y}}, T_2|_{\mathcal{Y}}) \quad (3)$$

holds by using a constructive argument. Let N be a hybridization network on $\mathcal{Y}$ that displays $T_1|_{\mathcal{Y}}$ and $T_2|_{\mathcal{Y}}$ and has exactly $h(T_1|_{\mathcal{Y}}, T_2|_{\mathcal{Y}})$ reticulate nodes. For the purpose of the upcoming construction, view the root of N as a node labeled ρ adjoined to the original root by a pendant edge. Further, let $\mathcal{X}_i$ be an element of the pendant partition $\mathcal{P}$ for T_1 with regards to $\mathcal{X} \setminus \mathcal{Y}$. Now, obtain a network N' from N by attaching $T_1|_{\mathcal{X}_i}$ to N by a new edge that joins the root of $T_1|_{\mathcal{X}_i}$ with a new node that subdivides an edge of N such that the resulting network displays $T_1|_{(\mathcal{L}(N) \cup \mathcal{X}_i)}$, where $\mathcal{L}(N)$ denotes the set of taxa that labels N. Since N displays $T_1|_{\mathcal{Y}}$ and since $T_1|_{\mathcal{X}_i}$ is, by definition, a maximal pendant subtree of T_1, note that this is always possible. By construction, as N does not contain any directed cycle, N' does not contain such a cycle either. Thus, N' is a hybridization network on $\mathcal{L}(N) \cup \mathcal{X}_i$ that displays $T_1|_{(\mathcal{L}(N) \cup \mathcal{X}_i)}$ and, trivially, $h(N) = h(N')$. Now, setting N to be N' and using a similar construction for each element in $\mathcal{P} \setminus \{\{\mathcal{X}_i\}\}$ and the pendant partition for T_2 with regards to $\mathcal{X} \setminus \mathcal{Y}$, Equation 3 follows. Combining Equations 2 and 3 establishes the theorem. □

ACKNOWLEDGMENTS

The authors would like to thank Stephan Koblmüller for helpful discussions related to the data set described in the fish application section.

REFERENCES

1. C. Semple and M. Steel, *Phylogenetics*. Oxford University Press, 2003.
2. J. Felsenstein, *Inferring Phylogenies*. Sinauer Associates, Inc., 2004.
3. D. H. Huson and D. Bryant, "Application of phylogenetic networks in evolutionary studies," *Molecular Biology and Evolution*, vol. 23, pp. 254–267, 2006.
4. D. H. Huson and C. Scornavacca, "A survey of combinatorial methods for phylogenetic networks," *Genome Biology and Evolution*, vol. 3, pp. 23–35, 2011. [Online]. Available: http://gbe.oxfordjournals.org/content/3/23.abstract
5. H.-J. Bandelt, P. Forster, and A. Röhl, "Median-joining networks for inferring intraspecific phylogenies," *Molecular Biology and Evolution*, vol. 16, pp. 37–48, 1999.
6. D. Bryant and V. Moulton, "Neighbor-net: An agglomerative method for the construction of phylogenetic networks," *Molecular Biology and Evolution*, vol. 21, no. 2, pp. 255–265, 2004.
7. P. J. Planet, S. C. Kachlany, D. H. Fine, R. DeSalle, and D. H. Figurski, "The widespread colonization island of *actinobacillus actinomycetemcomitans*," *Nature Genetics*, vol. 34, pp. 193 – 198, 2003.

8. S. Koblmueller, N. Duftner, K. Sefc, M. Aibara, M. Stipacek, M. Blanc, B. Egger, and C. Sturmbauer, "Reticulate phylogeny of gastropod-shell-breeding cichlids from lake tanganyika - the result of repeated introgressive hybridization," *BMC Evolutionary Biology*, vol. 7, no. 1, p. 7, 2007.
9. M. Bordewich and C. Semple, "Computing the minimum number of hybridisation events for a consistent evolutionary history," *Discrete Applied Mathematics*, vol. 155, no. 8, pp. 914–928, 2007.
10. S. Linz and C. Semple, "Hybridization in nonbinary trees," *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, vol. 6, no. 1, pp. 30–45, 2009.
11. J. Collins, S. Linz, , and S. C., "Quantifying hybridization in realistic time," *Journal of Computational Biology*, vol. 18, pp. 1305–1318, 2011.
12. Z.-Z. Chen, L. Wang, and S. Yamanaka, "A fast tool for minimum hybridization networks," *BMC bioinformatics*, vol. 13, no. 1, p. 155, 2012.
13. C. Whidden, R. Beiko, and N. Zeh, "Fast FPT algorithms for computing rooted agreement forests: theory and experiments," in *Experimental Algorithms*, ser. Lecture Notes in Computer Science. Springer, 2010, vol. 6049, pp. 141–153.
14. B. Albrecht, C. Scornavacca, A. Cenci, and D. Huson, "Fast computation of minimum hybridization networks," *Bioinformatics*, vol. 28, no. 2, pp. 191–197, 2012.
15. C. Whidden, R. G. Beiko, and N. Zeh, "Fixed-parameter algorithms for maximum agreement forests," *SIAM Journal on Computing*, vol. 42, no. 4, pp. 1431–1466, 2013.
16. T. Piovesan and S. Kelk, "A simple fixed parameter tractable algorithm for computing the hybridization number of two (not necessarily binary) trees," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 10, no. 1, pp. 18–25, 2013.
17. J. Chen, J.-H. Fan, and S.-H. Sze, "Parameterized and approximation algorithms for maximum agreement forest in multifurcating trees," *Theoretical Computer Science*, vol. 562, no. C, pp. 496–512, 2015.
18. C. Whidden, R. G. Beiko, and N. Zeh, "Fixed-parameter and approximation algorithms for maximum agreement forests of multifurcating trees," *arXiv.org*, May 2013.
19. Grass Phylogeny Working Group, "Phylogeny and subfamilial classification of the grasses (Poaceae)," *Annals of the Missouri Botanical Garden*, vol. 88, no. 3, pp. 373–457, 2001.
20. N. P. Tippery and D. H. Les, "Phylogenetic relationships and morphological evolution in nymphoides (menyanthaceae)," *Systematic Botany*, vol. 36, no. 4, pp. 1101–1113, 2011.
21. M. Baroni, S. Grünewald, V. Moulton, and C. Semple, "Bounding the number of hybridization events for a consistent evolutionary history," *Journal of Mathematical Biology*, vol. 51, pp. 171–182, 2005.
22. D. H. Huson, R. Rupp, and C. Scornavacca, *Phylogenetic Networks*. Cambridge University Press, 2010.
23. J. Mallet, "Hybridization, ecological races and the nature of species: empirical evidence for the ease of speciation," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 363, no. 1506, pp. 2971–2986, 2008.
24. M. L. Arnold, "Transfer and origin of adaptations through natural hybridization: were anderson and stebbins right?" *The Plant Cell Online*, vol. 16, no. 3, pp. 562–570, 2004.
25. J. Mallet, "Hybrid speciation," *Nature*, vol. 446, no. 7133, pp. 279–283, Mar. 2007.
26. S. Linz, "Reticulation in evolution," Ph.D. dissertation, Heinrich Heine University Düsseldorf, 2008.
27. D. H. Huson and R. Rupp, "Summarizing multiple gene trees using cluster networks," in *Algorithms in Bioinformatics, WABI 2008*, K. Crandall and J. Lagergren, Eds. Springer Berlin/Heidelberg, 2008, vol. 5251, pp. 296–305.
28. D. H. Huson and C. Scornavacca, "Dendroscope 3 - a program for computing and drawing rooted phylogenetic trees and networks," *Systematic Biology*, vol. 61, no. 6, pp. 1061–7, 2012.
29. J. Collins, "Rekernelisation algorithms in hybrid phylogenies," Master's thesis, University of Canterbury, 2009.
30. M. Bordewich, S. Linz, K. St. John, and C. Semple, "A reduction algorithm for computing the hybridization number of two trees," *Evolutionary Bioinformatics*, vol. 3, pp. 86–98, 2007.
31. Y. Wu, "Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees," *Bioinformatics*, vol. 26, no. 12, pp. i140–i148, 2010.
32. Y. Wu and J. Wang, "Fast computation of the exact hybridization number of two phylogenetic trees," *Bioinformatics Research and Applications*, pp. 203–214, 2010.
33. M. D. Pirie, A. M. Humphreys, N. P. Barker, and H. P. Linder, "Reticulation, data combination, and inferring evolutionary history: an example from Danthonioideae (Poaceae)," *Syst Biol*, vol. 58, no. 6, pp. 612–28, 2009.

Daniel H. Huson studied mathematics at Bielefeld University. He was a post-doc in Computer Science at the University of Pennsylvania and in Applied Math at Princeton University 1997-99. He then worked as a senior staff scientist at Celera Genomics in Gene Myers' group. Since 2002 he is Professor of Algorithms in Bioinformatics at the University of Tübingen. He is currently a Visiting Professor at the National University of Singapore. He works on algorithms and software for phylogenetics, genomics and metagenomics.

Simone Linz studied Biology and Computer Science at the University of Düsseldorf. She was awarded a 3-year Marie Curie International Outgoing Fellowship in 2011, which she spent in the School of Mathematics and Statistics at the University of Canterbury and the Center for Bioinformatics at the University of Tübingen. Since 2014, she is a lecturer in the Department of Computer Science at the University of Auckland. Her research interests are in phylogenetics, graph theory and computational complexity.