

# A Unifying View on Approximation and FPT of Agreement Forests (Extended Abstract<sup>\*</sup>)

Chris Whidden<sup>\*\*</sup> and Norbert Zeh<sup>\*\*\*</sup>

Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada  
{whidden,nzeh}@cs.dal.ca

**Abstract.** We provide a unifying view on the structure of maximum (acyclic) agreement forests of rooted and unrooted phylogenies. This enables us to obtain linear- or  $O(n \log n)$ -time 3-approximation and improved fixed-parameter algorithms for the subtree prune and regraft distance between two rooted phylogenies, the tree bisection and reconnection distance between two unrooted phylogenies, and the hybridization number of two rooted phylogenies.

## 1 Introduction

Phylogenies, or evolutionary trees, are a standard model to represent the evolutionary history of a set of species and are an indispensable tool in evolutionary biology [14]. Since determining the correct phylogeny of a set of species is hard, a host of heuristic methods for computing phylogenies have been proposed. The computation of distances between phylogenies under different metrics has proven essential for assessing the quality of phylogenies proposed by these heuristics, as well as for visualizing tree space (see, e.g., [13]). Of particular interest are metrics that model reticulation events, such as hybridization, lateral gene transfer, and recombination. These events result in species being composites of genes derived from different ancestors, and the analysis of different genes may produce different phylogenies over the same set of species. The *subtree prune and regraft* (rSPR) distance [11] and the *hybridization number* [2] of two rooted trees over the same set of species are important tools that often help to discover such events [15,16]. A related distance measure for unrooted trees is the *tree bisection and reconnection* (TBR) distance [1].

While biologically meaningful, TBR distance, rSPR distance, and hybridization number are NP-hard to compute [1, 6, 8]. As a result, significant efforts have been made to develop approximation and fixed-parameter (FPT) algorithms, as well as heuristic approaches [3, 12], for computing these distances. The best previous approximation algorithm for rSPR distance [17] provides a 3-approximation in  $O(n^2)$  time. This algorithm builds on earlier results from [11].

---

<sup>\*</sup> For details, see [18].

<sup>\*\*</sup> Supported by an NSERC CGS-M graduate scholarship.

<sup>\*\*\*</sup> Supported in part by NSERC and the Canada Research Chairs programme.

Another 3-approximation algorithm [5] has running time  $O(n^5)$ ; the “shifting lemma” central to its analysis is also the key to our new results. The same paper presents the best previous FPT algorithm for rSPR distance with running time  $O(4^k k^4 + n^3)$ , where  $k$  is the distance between the two trees. For TBR distance, the best previous approximation algorithm [9] computes an 8-approximation in polynomial time, and the best previous FPT algorithm [10] has running time  $O(4^k k^5 + p(n))$ , where  $p(\cdot)$  is a polynomial function. We are not aware of any approximation algorithms for hybridization number; the best previous FPT algorithm for this problem [7] has running time  $O((28k)^k + n^3)$ .

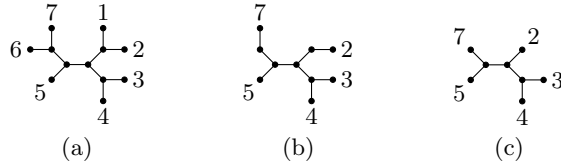
The contribution of this paper is to provide a unifying view on the structure of rooted and unrooted agreement forests, using the “shifting lemma” mentioned above. This allows us to show that the framework of the algorithms of [4, 11, 17] can be used not only to approximate rSPR distances but also to obtain approximation and FPT algorithms for rSPR distance, TBR distance, and hybridization number. Our approximation algorithms provide a 3-approximation of the distance between the two trees and run in linear or, in the case of hybridization number,  $O(n \log n)$  time. Our FPT algorithms for rSPR distance, TBR distance, and hybridization number have running times  $O(3^k n)$ ,  $O(4^k n)$ , and  $O(3^k n \log n)$ , respectively. Using standard kernelizations, their running times can be reduced to  $O(3^k k + n^3)$ ,  $O(4^k k + n^3)$ , and  $O(3^k k \log k + n^3)$ , respectively. All our algorithms represent improvements over the best previous algorithms for the same problems, substantial ones except in the case of the approximation algorithm for rSPR distance and the FPT algorithm for TBR distance. It should be noted here that “our” 3-approximation algorithm for rSPR distance is the algorithm of [17], with a minor modification to reduce its running time to  $O(n)$ . We believe, however, that the correctness proof obtained using our approach is simpler than the one presented in [17].

The rest of this paper is organized as follows. Section 2 introduces the necessary terminology and notation. Section 3 presents the main structural theorems at the heart of our approximation and FPT algorithms. Section 4 presents the approximation algorithms. Section 5 discusses briefly how to turn the approximation algorithms into FPT algorithms based on bounded search trees.

## 2 Preliminaries

Throughout this paper, we mostly use the definitions and notation from [1, 4–6, 17]. An *(unrooted binary phylogenetic) X-tree* is a tree  $T$  whose leaves are the elements of a label set  $X$  and whose internal nodes each have degree three. For a subset  $V$  of  $X$ ,  $T(V)$  is the smallest subtree of  $T$  that connects all nodes in  $V$ . The *V-tree induced by T* is the tree  $T|V$  obtained from  $T(V)$  using *forced contractions*, each of which replaces a vertex of degree two and its incident edges with a single edge between its neighbours.

A *rooted X-tree* is obtained from an unrooted one,  $T$ , by subdividing one of  $T$ ’s edges, declaring the node this introduces to be the root, and defining parent-child and ancestor-descendant relations accordingly. For a subset  $V$  of  $X$ ,  $T(V)$



**Fig. 1.** (a) An  $X$ -tree  $T$ . (b) The subtree  $T(V)$  for  $V = \{2, 3, 4, 5, 7\}$ . (c) The tree  $T|V$  obtained from  $T(V)$  using forced contractions.

and  $T|V$  are defined as in the unrooted case, but the construction of  $T|V$  from  $T(V)$  excludes the root of  $T(V)$  from forced contractions.

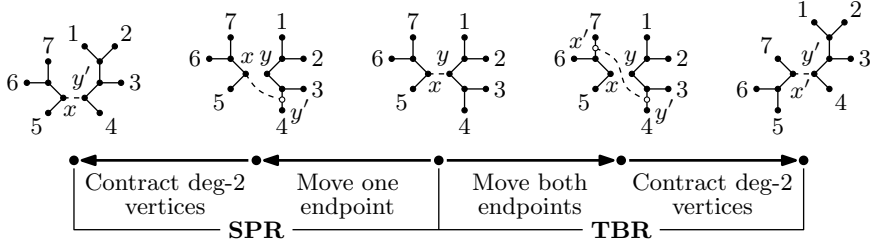
Given an unrooted  $X$ -tree  $T$ , a *tree bisection and reconnection* (TBR) operation cuts an edge  $xy$ , thereby dividing  $T$  into two subtrees  $T_x$  and  $T_y$  containing  $x$  and  $y$ , respectively. Then it introduces two new vertices  $x'$  and  $y'$  into  $T_x$  and  $T_y$  by subdividing one edge of each tree, and adds an edge  $x'y'$  to reconnect the two trees. Finally,  $x$  and  $y$  are removed using forced contractions. A *subtree prune and regraft* (SPR) operation is the one-sided equivalent of a TBR operation in that it introduces a new vertex  $y'$  into only  $T_y$ , adds an edge  $x'y'$ , and then removes  $y$  using a forced contraction. Figure 2 illustrates both operations. For a rooted tree  $T$ , a *rooted SPR* (rSPR) operation is an SPR operation in which the endpoints of edge  $xy$  are chosen so that  $T_y$  contains the root of  $T$ . Moreover, if  $y$  is the root of  $T_y$ , it is removed and its child becomes the new root.

TBR and rSPR operations define distance measures  $d_{\text{TBR}}(\cdot, \cdot)$  and  $d_{\text{rSPR}}(\cdot, \cdot)$  between  $X$ -trees, defined as the number of such operations required to transform one tree into the other. A related distance measure for rooted  $X$ -trees is their *hybridization number*,  $\text{hyb}(T_1, T_2)$ , which is defined in terms of hybrid networks of the two trees. A *hybrid network* of  $T_1$  and  $T_2$  is a directed acyclic graph  $H$  such that both  $T_1$  and  $T_2$ , with their edges directed away from the root, can be obtained from  $H$  by deleting edges and performing forced contractions. For a vertex  $x \in H$ , let  $\text{deg}_{\text{in}}(x)$  be its in-degree and  $\text{deg}_{\text{in}}^-(x) = \max(0, \text{deg}_{\text{in}}(x) - 1)$ . Then the hybridization number of  $T_1$  and  $T_2$  is  $\min_H \sum_{x \in H} \text{deg}_{\text{in}}^-(x)$ , where the minimum is taken over all hybrid networks  $H$  of  $T_1$  and  $T_2$ .

TBR distance, rSPR distance, and hybridization number are known to be one less than the sizes of appropriately defined maximum agreement forests (MAF's). To define these MAF's, we first introduce some terminology.

Given a forest  $F$  and a subset  $E$  of its edges,  $F - E$  denotes the forest obtained by deleting the edges in  $E$  from  $F$ . If  $F$  has components  $T_1, T_2, \dots, T_k$  with label sets  $X_1, X_2, \dots, X_k$ ,  $F$  yields the forest  $F'$  whose components  $T'_1, T'_2, \dots, T'_k$  satisfy  $T'_i = T_i|X_i$ , for all  $1 \leq i \leq k$ ; if all nodes of a component  $T_i$  are unlabelled (that is,  $X_i = \emptyset$ ), we define  $T_i|X_i = \emptyset$  and  $T'_i = \emptyset$ . If  $T - E$  yields  $F$ , for an  $X$ -tree  $T$  and a subset  $E$  of its edges, we say that  $F$  is a *forest of  $T$* .

Given two  $X$ -trees  $T_1$  and  $T_2$ , a forest  $F$  is an *agreement forest* of  $T_1$  and  $T_2$  if there exist edge sets  $E_1$  and  $E_2$  such that  $T_1 - E_1$  and  $T_2 - E_2$  yield  $F$ ; see Figure 3.  $F$  is a *maximum agreement forest* (MAF) of  $T_1$  and  $T_2$  if there is no agreement forest of  $T_1$  and  $T_2$  with fewer connected components. We use



**Fig. 2.** Illustration of TBR and SPR operations.

$m(T_1, T_2)$  to denote the number of connected components in an MAF of  $T_1$  and  $T_2$ . For a forest  $F$  of  $T_2$ , we use  $e(T_1, T_2, F)$  to denote the size of the smallest edge set  $E$  such that  $F - E$  yields an agreement forest of  $T_1$  and  $T_2$ . Allen and Steel [1] showed that, for two unrooted  $X$ -trees,  $T_1$  and  $T_2$ ,  $d_{\text{TBR}}(T_1, T_2) = e(T_1, T_2, T_2) = m(T_1, T_2) - 1$ .

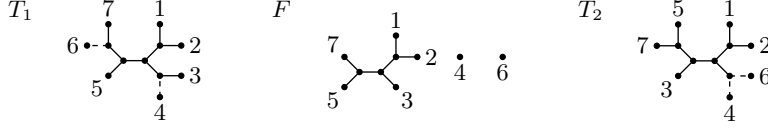
In the rooted case, MAF's are similarly related to rSPR distances. This, however, is true only if the MAF is defined with respect to augmented versions of the two trees, obtained by adding a new root node with label  $\rho$  to both trees and making it the parent of their original roots. An agreement forest of two rooted  $X$ -trees  $T_1$  and  $T_2$  is then defined as a collection  $\{T'_\rho, T'_1, T'_2, \dots, T'_k\}$  of rooted trees with label sets  $X_\rho, X_1, X_2, \dots, X_k$  that satisfy the following conditions [6]:

1. The label sets  $X_\rho, X_1, X_2, \dots, X_k$  partition  $X \cup \{\rho\}$ , and  $\rho \in X_\rho$ .
2. For all  $i \in \{\rho, 1, 2, \dots, k\}$ ,  $T'_i = T_1|X_i = T_2|X_i$  in the rooted sense.
3. The trees in each of the sets  $\{T_1(X_i) \mid i \in \{\rho, 1, 2, \dots, k\}\}$  and  $\{T_2(X_i) \mid i \in \{\rho, 1, 2, \dots, k\}\}$  are vertex-disjoint.

An MAF is again one with the minimum number of connected components. Bordewich and Semple [6] proved that, for two rooted  $X$ -trees  $T_1$  and  $T_2$ ,  $d_{\text{rSPR}}(T_1, T_2) = e(T_1, T_2, T_2) = m(T_1, T_2) - 1$ .

The hybridization number of two rooted  $X$ -trees  $T_1$  and  $T_2$  corresponds to an MAF of  $T_1$  and  $T_2$  with an additional constraint. An agreement forest  $F$  of  $T_1$  and  $T_2$  is said to contain a cycle if there exist two nodes  $x$  and  $y$  that are roots of trees in  $F$  and such that  $x$  is an ancestor of  $y$  in  $T_1$ , while  $y$  is an ancestor of  $x$  in  $T_2$ . (Each node  $x$  in  $F$  can be mapped to nodes  $\phi_1(x)$  in  $T_1$  and  $\phi_2(y)$  in  $T_2$  by defining  $X_x$  to be the set of labelled descendants of  $x$  in  $F$  and defining  $\phi_i(x)$  to be the lowest common ancestor in  $T_i$  of all nodes in  $X_x$ .)  $F$  is *acyclic* if it contains no cycles. A *maximum acyclic agreement forest* (MAAF) of  $T_1$  and  $T_2$  is an agreement forest with the minimum number of connected components among all acyclic agreement forests of  $T_1$  and  $T_2$ . We denote its size by  $\tilde{m}(T_1, T_2)$  and the number of edges in a forest  $F$  of  $T_2$  that must be cut to obtain an acyclic agreement forest of  $T_1$  and  $T_2$  by  $\tilde{e}(T_1, T_2, F)$ . Baroni et al. [2] showed that, for two rooted  $X$ -trees  $T_1$  and  $T_2$ ,  $\text{hyb}(T_1, T_2) = \tilde{e}(T_1, T_2, T_2) = \tilde{m}(T_1, T_2) - 1$ .

For two nodes  $a$  and  $b$  of a forest  $F$ , we write  $a \sim_F b$  to indicate that there exists a path from  $a$  to  $b$  in  $F$ . Two labelled leaves  $a$  and  $c$  with the same



**Fig. 3.** Two  $X$ -trees  $T_1$  and  $T_2$  and an agreement forest  $F$  of  $T_1$  and  $T_2$ .  $F$  is obtained from each tree by cutting the dashed edges.

neighbour, denoted  $r_{ac}$ , form a *sibling pair*  $(a, c)$ . Our algorithms rely on the following two lemmas, which were proved in [4, 6] for rooted MAF's but are easily seen to apply also to the unrooted case.

**Lemma 1.** *Let  $T_1$  and  $T_2$  be  $X$ -trees,  $F$  a forest of  $T_2$ , and  $(a, c)$  a sibling pair that exists in  $T_1$  and  $F$ . Let  $T'_1$ ,  $T'_2$ , and  $F'$  be obtained from  $T_1$ ,  $T_2$ , and  $F$  by relabelling node  $a$  as  $(a, c)$ , removing  $c$ , and performing forced contractions to eliminate degree-2 nodes. We refer to this as contracting the sibling pair  $(a, c)$ . Then  $e(T_1, T_2, F) = e(T'_1, T'_2, F')$ .*

**Lemma 2.** *Let  $T_1$  and  $T_2$  be  $X$ -trees,  $F$  a forest of  $T_2$ , and  $c$  a singleton in  $F$ . Let  $T'_1$ ,  $T'_2$ , and  $F'$  be obtained from  $T_1$ ,  $T_2$ , and  $F$  by removing  $c$  and performing forced contractions to eliminate degree-2 nodes. Then  $e(T_1, T_2, F) = e(T'_1, T'_2, F')$ .*

### 3 The Structure of Agreement Forests

This section presents results that provide the intuition and correctness proofs for our approximation and FPT algorithms, presented in Sections 4 and 5. All these algorithms start with a pair of trees  $(T_1, T_2)$  and then cut edges, remove singletons, and contract sibling pairs in both trees until they are identical. The intermediate state is that  $T_1$  has been reduced to a smaller tree, and  $T_2$  to a forest  $F$ . Each iteration has to decide which edges in  $F$  to cut next. The results in this section identify small edge sets in  $F$  so that at least one edge in each of these sets has the property that cutting it reduces  $e(T_1, T_2, F)$  by one. Thus, the approximation algorithm cuts all edges in the identified set, and the size of the edge set cut in each step gives the approximation ratio of the algorithm. The FPT algorithm tries each edge in the set in turn, and the size of the set gives the branching factor for a bounded search tree algorithm. The following lemma by Bordewich et al. [5] is the central tool used in all our proofs.

**Lemma 3 (Shifting Lemma).** *Let  $F$  be a forest of an  $X$ -tree,  $e$  and  $f$  edges in the same component of  $F$ , and  $E$  a subset of edges of  $F$  such that  $f \in E$  and  $e \notin E$ . Let  $v_f$  be the end-vertex of  $f$  closest to  $e$ , and  $v_e$  an end-vertex of  $e$ . If  $v_f \sim_{F-E} v_e$  and  $x \approx_{F-(E \cup \{e\})} v_f$  for all  $x \in X$ , then  $F - E$  and  $F - (E \setminus \{f\} \cup \{e\})$  yield the same forest.<sup>1</sup>*

<sup>1</sup> In the rooted case, it is assumed that  $\rho \in X$ .

The other tool we need is an observation that relates incompatible triples and quartets to agreement forests. A *triple*  $ab|c$  in a rooted tree  $T$  is defined by three leaves  $a, b, c$  such that the path from  $a$  to  $b$  is vertex-disjoint from the path from  $c$  to the root. A *quartet*  $ab|cd$  in an unrooted tree  $T$  is defined by four leaves  $a, b, c, d$  such that the two paths from  $a$  to  $b$  and from  $c$  to  $d$  are vertex-disjoint. Given a tree  $T$  and a forest  $F$ , we say a triple  $ab|c$  or quartet  $ab|cd$  of  $T$  is *incompatible* with  $F$  if its leaves do not all belong to the same component of  $F$  or define a different triple or quartet in  $F$  (e.g.,  $ac|b$  or  $ac|bd$ ).

- Observation 1.** (i) Let  $T_1$  and  $T_2$  be rooted  $X$ -trees,  $F$  a forest of  $T_2$ , and  $E$  a set of edges such that  $F - E$  yields an agreement forest of  $T_1$  and  $T_2$ . If  $ab|c$  is a triple of  $T_1$  incompatible with  $F$ , then  $a \approx_{F-E} b$  or  $a \approx_{F-E} c$ .
- (ii) Let  $T_1$  and  $T_2$  be unrooted  $X$ -trees,  $f$  a forest of  $T_2$ , and  $E$  a set of edges such that  $F - E$  yields an agreement forest of  $T_1$  and  $T_2$ . If  $ab|cd$  is a quartet of  $T_1$  incompatible with  $F$ , then  $a \approx_{F-E} b$ ,  $a \approx_{F-E} c$  or  $c \approx_{F-E} d$ .

Now consider two  $X$ -trees  $T_1$  and  $T_2$  and a forest  $F$  of  $T_2$ , and let  $(a, c)$  be a sibling pair of  $T_1$  that does not exist in  $F$  and such that neither  $a$  nor  $c$  is a singleton in  $F$ . If  $a$  and  $c$  belong to the same tree of  $F$ , the *sibling*  $b$  of  $a$  in  $F$  is the node adjacent to  $a$ 's neighbour that is not on the path from  $a$  to  $c$  in  $F$ ; otherwise,  $b$  is any node at distance two from  $a$  in  $F$ . Note that  $b$  may not be a leaf. We use  $e_a$  and  $e_b$  to denote the edges connecting  $a$  and  $b$  to their common neighbour  $r_{ab}$ , and  $B$  to denote the subtree of  $F$  induced by all nodes  $b'$  such that  $e_b$  belongs to the path from  $b'$  to  $a$ . The sibling  $d$  of  $c$  and edges  $e_c$  and  $e_d$  are defined analogously. In the rooted case, we choose  $b$  so that  $r_{ab}$  is the parent of  $a$  and  $b$ ; to ensure that  $c \notin B$ , we assume that the distance from the root to  $a$  is no less than the distance from the root to  $c$ , which is easily ensured by swapping the roles of  $a$  and  $c$  if necessary.

With these tools in hand, we are now ready to prove three results characterizing edges that need to be cut in order to make progress towards an M(A)AF. The first result considers rooted MAF's.

**Theorem 1.** Let  $T_1$  and  $T_2$  be rooted  $X$ -trees,  $F$  a forest of  $T_2$ , and  $(a, c)$  a sibling pair of  $T_1$  that is not a sibling pair of  $F$ . Assume that neither  $a$  nor  $c$  is a singleton in  $F$ . Then  $e(T_1, T_2, F - \{e_x\}) = e(T_1, T_2, F) - 1$ , for some  $x \in \{a, b, c\}$ . In particular,  $e(T_1, T_2, F - \{e_a, e_b, e_c\}) \leq e(T_1, T_2, F) - 1$ .

*Proof.* It suffices to prove that there exists an edge set  $E$  of size  $e(T_1, T_2, F)$  such that  $F - E$  yields an MAF of  $T_1$  and  $T_2$  and  $E \cap \{e_a, e_b, e_c\} \neq \emptyset$ . So assume that  $F - E$  yields an MAF  $F'$  of  $T_1$  and  $T_2$  and that  $E \cap \{e_a, e_b, e_c\} = \emptyset$ . We prove that we can replace an edge  $f \in E$  with an edge in  $\{e_a, e_b, e_c\}$  without changing the forest yielded by  $F - E$ .

First assume that  $b' \approx_{F-E} r_{ab}$ , for all leaves  $b' \in B$ . In this case, we choose an arbitrary leaf  $b' \in B$  and the first edge  $f \in E$  on the path from  $r_{ab}$  to  $b'$ . Lemma 3 implies that  $F - E$  and  $F - (E \setminus \{f\} \cup \{e_b\})$  yield the same forest.

Now assume that there exists a leaf  $b' \in B$  such that  $b' \sim_{F-E} r_{ab}$ . We prove that this implies that  $c$  is a singleton in  $F'$ . Since  $e_a \notin E$ , we have  $a \sim_{F-E} r_{ab}$

and, hence,  $a \sim_{F-E} b'$ . Since  $(a, c)$  is a sibling pair of  $T_1$ ,  $ac|b'$  is a triple of  $T_1$ , while  $c \notin B$  implies that either  $ab'|c$  is a triple of  $F$  or  $a \not\sim_F c$ . In either case, the triple  $ac|b'$  is incompatible with  $F$ . By Observation 1(i), this implies that  $a \not\sim_{F-E} c$  because  $F-E$  yields an agreement forest of  $T_1$  and  $T_2$  and  $a \sim_{F-E} b'$ . Then, however, either  $a$  or  $c$  is a singleton of  $F'$  because  $(a, c)$  is a sibling pair of  $T_1$ . Since  $a \sim_{F-E} b'$ ,  $a$  cannot be a singleton of  $F'$ , that is,  $c$  must be a singleton.

Now, as  $c$  is not a singleton in  $F$ , there exists a leaf  $l$  such that  $c \sim_F l$ . Since  $c$  is a singleton in  $F'$ , at least one of the edges on the path from  $c$  to  $l$  in  $F$  belongs to  $E$ ; let  $f$  be the one closest to  $c$ . Since  $c$  is a singleton in  $F'$ , edges  $e_c$  and  $f$  satisfy the conditions of Lemma 3, and  $F-E$  and  $F-(E \setminus \{f\} \cup \{e_c\})$  yield the same forest.  $\square$

Note that Theorem 1 also holds if we replace  $e(\cdot, \cdot, \cdot)$  with  $\tilde{e}(\cdot, \cdot, \cdot)$ . To see this, it suffices to consider a set  $E$  in the proof such that  $F-E$  yields an MAAF instead of an MAF. The next theorem provides an analogous result for unrooted MAF's. Its proof is similar to that of Theorem 1 and is omitted.

**Theorem 2.** *Let  $T_1$  and  $T_2$  be unrooted  $X$ -trees,  $F$  a forest of  $T_2$ , and  $(a, c)$  a sibling pair of  $T_1$  that is not a sibling pair of  $F$ . Assume that neither  $a$  nor  $c$  is a singleton in  $F$ . Then  $e(T_1, T_2, F - \{e_x\}) = e(T_1, T_2, F) - 1$ , for some  $x \in \{a, b, c, d\}$ .*

Similar to Theorem 1, Theorem 2 implies that  $e(T_1, T_2, F - \{e_a, e_b, e_c, e_d\}) \leq e(T_1, T_2, F) - 1$ . However, we can do a little better.

**Theorem 3.** *Let  $T_1$  and  $T_2$  be unrooted  $X$ -trees,  $F$  a forest of  $T_2$ , and  $(a, c)$  a sibling pair of  $T_1$  that is not a sibling pair of  $F$ . Assume that neither  $a$  nor  $c$  is a singleton in  $F$ . Then  $e(T_1, T_2, F - \{e_a, e_b, e_c\}) \leq e(T_1, T_2, F) - 1$ .*

*Proof.* Let  $E$  be an edge set of size  $e(T_1, T_2, F)$  such that  $F-E$  yields an MAF of  $T_1$  and  $T_2$ . We can again assume that  $E \cap \{e_a, e_b, e_c\} = \emptyset$ , as otherwise the theorem holds trivially. As in the proof of Theorem 1, we can also assume there exists a leaf  $b' \in B$  such that  $b' \sim_{F-E} r_{ab}$  and, hence,  $b' \sim_{F-E} a$ . We prove that this implies that  $c$  is a singleton of the forest  $F'$  yielded by  $F - E_{ab}$ , where  $E_{ab} = E \cup \{e_a, e_b\}$ . As in the proof of Theorem 1, this implies that there exists an edge  $f \in E$  such that  $F - E_{ab}$  and  $F - (E_{ab} \setminus \{f\} \cup \{e_c\})$  yield the same forest,  $F'$ , which proves the theorem. We distinguish two cases.

If  $a \not\sim_{F-E} c$ ,  $c$  is a singleton of  $F'$  because  $(a, c)$  is a sibling pair of  $T_1$  and  $a \sim_{F-E} b'$ . If  $a \sim_{F-E} c$ ,  $(a, c)$  being a sibling pair in  $T_1$  and  $c \notin B$  imply that  $ac|b'd'$  is a quartet of  $T_1$  incompatible with  $F$ , for all  $d' \notin X_B \cup \{a, c\}$ , where  $X_B$  is the label set of  $B$ . Hence, by Observation 1(ii),  $a \sim_{F-E} b'$  and  $a \sim_{F-E} c$  imply that  $c \not\sim_{F-E} d'$ , for each such leaf  $d'$ . This, however, implies that  $c \not\sim_{F-E_{ab}} x$ , for all  $x \in X$ , that is,  $c$  is a singleton of  $F'$ .  $\square$

While Theorem 1 suffices as a basis for an algorithm to compute or approximate an MAF of two rooted trees, a little extra work is required to obtain an MAAF. As observed after the proof of Theorem 1, we can use this theorem to

make progress towards an MAAF until we obtain an agreement forest of the two trees. If this forest is in fact acyclic, we are done. Otherwise, we need to continue cutting edges to remove all cycles that may exist. The next theorem identifies candidate edges to cut. In this theorem, we consider two trees,  $A$  and  $B$ , of the agreement forest whose roots,  $a$  and  $b$ , form a cycle. We call  $(a, b)$  a *cycle pair* and use  $e_a$  to denote any of the two edges in  $A$  incident to  $a$ , and  $e_b$  to denote any of the two edges in  $B$  incident to  $b$ .

**Theorem 4.** *Let  $T_1$  and  $T_2$  be two rooted  $X$ -trees,  $F$  an agreement forest of  $T_1$  and  $T_2$ , and  $(a, b)$  a cycle pair of  $F$ . Then  $\tilde{e}(T_1, T_2, F - \{e_x\}) = \tilde{e}(T_1, T_2, F) - 1$ , for some  $x \in \{a, b\}$ . In particular,  $\tilde{e}(T_1, T_2, F - \{e_a, e_b\}) \leq \tilde{e}(T_1, T_2, F) - 1$ .*

*Proof.* Once again, our goal is to show that there exists a set  $E$  of  $\tilde{e}(T_1, T_2, F)$  edges of  $F$  such that  $F - E$  yields an MAAF,  $F'$ , of  $T_1$  and  $T_2$  and  $E \cap \{e_a, e_b\} \neq \emptyset$ . So we choose  $E$  to be a set such that  $F - E$  yields an MAAF  $F'$  of  $T_1$  and  $T_2$ , and we show that, if  $E \cap \{e_a, e_b\} = \emptyset$ , we can find an edge  $f \in E$  such that, for some  $x \in \{a, b\}$ ,  $F - E$  and  $F - (E \setminus \{f\} \cup \{e_x\})$  yield the same forest. Let  $A_1$  and  $A_2$  be the two subtrees of  $A$  rooted in  $a$ 's children, and let  $B_1$  and  $B_2$  be the two subtrees of  $B$  rooted in  $b$ 's children.

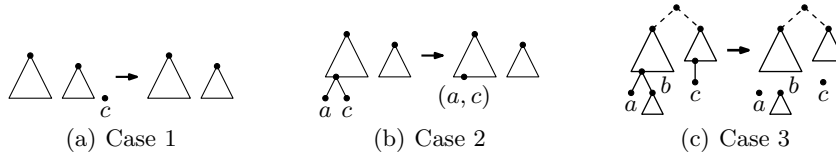
First observe that there exists an index  $i$  such that either  $a' \sim_{F-E} a$  for all  $a' \in X_{A_i}$  or  $b' \sim_{F-E} b$  for all  $b' \in X_{B_i}$ . If this was not the case, there would exist leaves  $a_1 \in A_1$ ,  $a_2 \in A_2$ ,  $b_1 \in B_1$ , and  $b_2 \in B_2$  such that  $a_1 \sim_{F-E} a_2$  and  $b_1 \sim_{F-E} b_2$ , implying that both  $a$  and  $b$  exist in  $F'$ , and  $F'$  would not be acyclic.

So assume w.l.o.g. that  $a' \sim_{F-E} a$ , for all  $a' \in A_1$ . In this case, Lemma 3 implies that, if we choose a leaf  $a' \in A_1$  and the edge  $f \in E$  closest to  $a$  on the path from  $a$  to  $a'$ , then  $F - E$  and  $F - (E \setminus \{f\} \cup \{e_a\})$  yield the same forest.  $\square$

## 4 Approximation Algorithms

*Rooted and Unrooted MAF.* The first algorithm we present is a 3-approximation algorithm for rooted MAF, that is, rSPR distance. This is essentially the algorithm discussed in [17], modified to achieve linear time. We include it here to demonstrate that Theorem 1 proves its correctness, and also as a reference for the other algorithms. The algorithm maintains a triple  $(T_1, T_2, F)$  and modifies it through a series of transformations. Initially,  $F = T_2$ .  $T_1$  and  $T_2$  shrink over time but are always trees with the same label set;  $F$  is always a forest of  $T_2$ . The algorithm also maintains a counter,  $D$ , of the number of edges in  $F$  it has cut so far. We use  $T_1^{(i)}$ ,  $T_2^{(i)}$ ,  $F^{(i)}$ , and  $D^{(i)}$  to denote snapshots of  $T_1$ ,  $T_2$ ,  $F$ , and  $D$  after the  $i$ th transformation. The algorithm terminates when the label set of  $T_1^{(i)}$  and  $T_2^{(i)}$  has size at most 2, including the root label  $\rho$ , which is never eliminated. The output is the value of  $D^{(i)}$  at the time of termination. Each iteration applies one of the following cases, illustrated in Figure 4.

1. As long as  $F$  contains a singleton  $c \neq \rho$ , the algorithm removes  $c$  from  $T_1$ ,  $T_2$ , and  $F$  and performs forced contractions in  $T_1$  and  $T_2$  to merge the other two edges incident to  $c$ 's parents in  $T_1$  and  $T_2$ .  $D$  remains unchanged.



**Fig. 4.** The three cases of the approximation algorithm for rooted MAF. Only  $F$  is shown. In Figure (c), the dashed edges indicate that  $a$  and  $c$  may or may not belong to the same tree of  $F$  in Case 3.

For the other two cases, the algorithm chooses a fixed sibling pair  $(a, c)$  of  $T_1$ .

2. If  $(a, c)$  is also a sibling pair of  $F$ , the algorithm contracts the sibling pair as discussed in Lemma 1.  $D$  remains unchanged.
3. If  $(a, c)$  is not a sibling pair in  $F$ , then assume w.l.o.g. that  $a$ 's distance from the root of  $T_2$  is no less than that of  $c$ . Node  $a$  must have a sibling  $b$  in  $F$  because  $F$  contains no singletons. In this case, the algorithm cuts edges  $e_a$ ,  $e_b$ , and  $e_c$  in  $F$  and increases  $D$  by three.  $T_1$  and  $T_2$  remain unchanged.

**Theorem 5.** *Given two rooted  $X$ -trees  $T_1$  and  $T_2$ , a 3-approximation of  $e(T_1, T_2, T_2) = d_{\text{rSPR}}(T_1, T_2)$  can be computed in linear time.*

*Proof.* We use the algorithm above and output the final value of  $D$  as the approximation of  $e(T_1, T_2, T_2)$ . We argue below that the algorithm terminates, in linear time. If the algorithm terminates after  $k$  iterations,  $k'$  of which change  $D$ , then its output is  $D^{(k)} = 3k'$ . We prove that  $e(T_1, T_2, T_2) \leq 3k' \leq 3e(T_1, T_2, T_2)$ , thereby proving that the value  $D^{(k)}$  is a 3-approximation of  $e(T_1, T_2, T_2) = d_{\text{rSPR}}(T_1, T_2)$ .

For every iteration that leaves  $D$  unchanged (Cases 1 and 2), Lemmas 1 and 2 show that the applied transformations do not alter  $e(T_1, T_2, F)$ , and thus  $e(T_1^{(i)}, T_2^{(i)}, F^{(i)}) = e(T_1^{(i-1)}, T_2^{(i-1)}, F^{(i-1)})$ . Every iteration that changes  $D$  applies Case 3. Since  $(a, c)$  is not a sibling pair of  $F$  in this case, and neither  $a$  nor  $c$  is a singleton in  $F$ , Theorem 1 implies that  $e(T_1^{(i)}, T_2^{(i)}, F^{(i)}) \leq e(T_1^{(i-1)}, T_2^{(i-1)}, F^{(i-1)}) - 1$ . Hence, we have  $e(T_1, T_2, T_2) \geq k'$ , that is,  $D^{(k)} = 3k' \leq 3e(T_1, T_2, T_2)$ . Conversely, since the  $3k'$  edges we cut in  $T_2$  yield an agreement forest of  $T_1$  and  $T_2$ , we have  $e(T_1, T_2, T_2) \leq 3k'$ .

To bound the running time of the algorithm, we observe that it terminates after  $O(n)$  iterations, as each iteration removes at least one vertex or edge from  $T_1$  or  $F$ . Using arguments similar to the ones presented in [4], we can show that each iteration takes constant time. (See [18] for details.)  $\square$

The 3-approximation algorithm for unrooted MAF and, hence, for TBR distance is the same as for the rooted case, except that the edges  $e_a$ ,  $e_b$ , and  $e_c$  in Case 3 are used in their unrooted meaning. Moreover, in the unrooted case, edge  $e_b$  is less trivial to identify in constant time. Instead, we cut  $e_a$  and one additional edge incident to  $r_{ab}$ . It is not hard to see that cutting any two of the three edges incident to  $r_{ab}$  has the same effect as cutting  $e_a$  and  $e_b$ . Hence, Theorem 3 establishes the correctness of this procedure, and we obtain:

**Theorem 6.** *Given two unrooted  $X$ -trees  $T_1$  and  $T_2$ , a 3-approximation of  $e(T_1, T_2, T_2) = d_{TBR}(T_1, T_2)$  can be computed in linear time.*

*Rooted MAAF.* The approximation algorithm for rooted MAAF consists of three phases: a preprocessing phase and two cutting phases. The preprocessing phase labels every node in  $T_1$  and  $T_2$  with its preorder number and with the interval of preorder numbers of its descendants, in order to identify cycles in an agreement forest of  $T_1$  and  $T_2$  later. The first cutting phase runs the algorithm for rooted MAF to obtain an agreement forest  $F$  of  $T_1$  and  $T_2$ . The second cutting phase identifies and breaks cycles in  $F$ . Whenever we cut an edge in either of the two cutting phases, we increase  $D$  by one. The algorithm terminates when no cycle pair remains in  $F$ . The output again is the final value of  $D$ .

To implement the second cutting phase, we maintain two sets,  $R_d$  and  $R_t$ , of roots of trees in  $F$ .  $R_d$  contains a set of roots that do not form any cycles with each other.  $R_t$  contains roots that may be involved in cycles. Initially,  $R_d = \emptyset$  and  $R_t$  contains the roots of all trees in  $F$ . Each iteration of the algorithm removes a root  $a$  from  $R_t$  and tests whether  $a$  forms a cycle with any root  $b \in R_d$ . This is true if and only if  $a$ 's preorder interval in  $T_1$  contains  $b$ 's, and  $b$ 's preorder interval in  $T_2$  contains  $a$ 's (or vice versa). If not, we add  $a$  to  $R_d$  and move on to the next root in  $R_t$ . If there is a root  $b \in R_d$  such that  $(a, b)$  is a cycle pair, we cut one of the two edges incident to each of  $a$  and  $b$  and increase  $D$  by two. This breaks the two trees in  $F$  with roots  $a$  and  $b$  into two subtrees each; their roots are the children of  $a$  and  $b$  in  $F$ . We add these children to  $R_t$  and then move on to the next iteration. The algorithm terminates when  $R_t = \emptyset$ , at which point  $F$  is an acyclic agreement forest of  $T_1$  and  $T_2$ .

Each iteration of this algorithm is easily implemented in linear time, resulting in a total running time of  $O(n^2)$ . In the full paper (see also [18]), we show how to reduce the running time to  $O(n \log n)$ . Thus, we have the following result.

**Theorem 7.** *Given two unrooted  $X$ -trees  $T_1$  and  $T_2$ , a 3-approximation of  $\tilde{e}(T_1, T_2, T_2) = \text{hyb}(T_1, T_2)$  can be computed in  $O(n \log n)$  time.*

*Proof.* We have already discussed the running time of the algorithm. To prove the approximation bound, consider all iterations over the two cutting phases of the algorithm. An iteration that leaves  $D$  unchanged does not increase  $\tilde{e}(T_1, T_2, T_2)$ . By Theorem 1, every application of Case 3 in the first cutting phase decreases  $\tilde{e}(T_1, T_2, F)$  by at least one and cuts three edges in  $F$ . By Theorem 4, every time we cut two edges in the current agreement forest  $F$  in the second cutting phase,  $\tilde{e}(T_1, T_2, F)$  decreases by at least one. Hence, the number  $k'$  of iterations that change  $D$  is at most  $\tilde{e}(T_1, T_2, T_2)$ , and each such iteration increases  $D$  by at most three. Thus,  $D \leq 3k' \leq 3\tilde{e}(T_1, T_2, T_2)$  at the end of the algorithm. On the other hand, once the algorithm terminates,  $R_t$  is empty, and the roots in  $R_d$  do not form cycles. The resulting agreement forest is therefore acyclic, and we cut  $D$  edges to obtain it. Thus,  $\tilde{e}(T_1, T_2, T_2) \leq D$ . Together with the upper bound, this shows that the final value of  $D$  is a 3-approximation of  $\tilde{e}(T_1, T_2, T_2)$ .  $\square$

## 5 Fixed-Parameter Algorithms

The approximation algorithms discussed in the previous section are easily modified to obtain fixed-parameter algorithms for the respective problems. As is customary when discussing such algorithms, we focus on the decision version: “Given two  $X$ -trees  $T_1$  and  $T_2$ , a distance measure  $d(\cdot, \cdot)$ , and a parameter  $k$ , is  $d(T_1, T_2) \leq k$ ?” If this decision version can be solved in  $O(c^k \text{poly}(n))$  time, then the exact distance  $d = d(T_1, T_2)$  can be found in  $O(c^d \text{poly}(n))$  time by iteratively trying larger guesses of  $k$  until we obtain the first positive answer.

To obtain such a decision algorithm for (rooted or unrooted) MAF, we modify the approximation algorithm from Section 4. We denote an invocation of the algorithm on trees  $T'_1, T'_2$ , forest  $F'$ , and distance bound  $k$  by  $\mathcal{A}(T'_1, T'_2, F', k)$ . If  $T'_1$  and  $T'_2$  have at most two nodes each, the algorithm returns “yes” if and only if  $k \geq 0$ . Otherwise, whenever the approximation algorithm applies Case 1 or 2, so does  $\mathcal{A}(T'_1, T'_2, F', k)$ . When the approximation algorithm would apply Case 3,  $\mathcal{A}(T'_1, T'_2, F', k)$  recurses. In the rooted case, the algorithm makes three recursive calls  $\mathcal{A}(T'_1, T'_2, F' - \{e_x\}, k - 1)$ , for  $x \in \{a, b, c\}$ , and returns “yes” if and only if one of these recursive calls does. In the unrooted case, the algorithm makes four recursive calls  $\mathcal{A}(T'_1, T'_2, F' - \{e_x\}, k - 1)$ , for  $x \in \{a, b, c, d\}$ , and again returns “yes” if and only if one of these recursive calls does.

**Theorem 8.** *For two rooted  $X$ -trees  $T_1$  and  $T_2$  and a parameter  $k$ , it takes  $O(3^k n)$  time to decide whether  $e(T_1, T_2, T_2) \leq k$ . In the unrooted case, it takes  $O(4^k n)$  time.*

*Proof.* The correctness of the algorithm follows immediately from Lemmas 1 and 2, and from Theorems 1 and 2. As for the running time, we can view each recursive call as a truncated invocation of the approximation algorithm from Section 4 that recurses as soon as it would invoke Case 3. Hence, each recursive call takes  $O(n)$  time. The recursion depth is  $k$ , and each recursive call spawns at most three recursive calls, four in the unrooted case. Hence, the number of recursive calls is  $O(3^k)$  in the rooted case and  $O(4^k)$  in the unrooted case. This gives the claimed running times of  $O(3^k n)$  and  $O(4^k n)$ , respectively.  $\square$

To obtain an FPT algorithm for MAAF, we augment the above algorithm for rooted MAF as follows. For every recursive call  $\mathcal{A}(T'_1, T'_2, F', k)$  that would output “yes” without recursing, we compute the corresponding agreement forest  $F$  of the two original trees  $T_1$  and  $T_2$ . Note that  $F$  is not necessarily an MAF, as  $k$  may be greater than  $e(T_1, T_2, T_2)$ . If  $F$  is acyclic, the algorithm answers “yes”. Otherwise, it invokes a second recursive algorithm  $\mathcal{B}(T_1, T_2, F, k)$ . This invocation returns “yes” if  $k \geq 0$  and  $F$  contains no cycle, and “no” if  $k < 0$ . If  $k \geq 0$  and  $F$  contains a cycle pair  $(a, b)$ ,  $\mathcal{B}(T_1, T_2, F, k)$  makes two recursive calls  $\mathcal{B}(T_1, T_2, F - \{e_a\}, k - 1)$  and  $\mathcal{B}(T_1, T_2, F - \{e_b\}, k - 1)$ , and returns “yes” if and only if one of the two calls does. The correctness and running time of the algorithm are established similarly to Theorem 8 (see [18]). Hence, we have:

**Theorem 9.** *For two rooted  $X$ -trees  $T_1$  and  $T_2$  and a parameter  $k$ , it takes  $O(3^k n \log n)$  time to decide whether  $\tilde{e}(T_1, T_2, T_2) \leq k$ .*

Known kernelizations [1, 6, 7] reduce trees  $T_1$  and  $T_2$  to trees  $T'_1$  and  $T'_2$  in  $O(n^3)$  time such that  $e(T_1, T_2, T_2) = e(T'_1, T'_2, T'_2) = k$  and  $T'_1$  and  $T'_2$  have size  $O(k)$ . The same holds for  $\tilde{e}(T_1, T_2, T_2)$ . Hence, we obtain the following corollary.

**Corollary 1.** *For two rooted  $X$ -trees  $T_1$  and  $T_2$  and a parameter  $k$ , it takes  $O(3^k k + n^3)$  time to decide whether  $e(T_1, T_2, T_2) \leq k$  and  $O(3^k k \log k + n^3)$  time to decide whether  $\tilde{e}(T_1, T_2, T_2) \leq k$ . In the unrooted case, it takes  $O(4^k k + n^3)$  time to decide whether  $e(T_1, T_2, T_2) \leq k$ .*

## References

1. B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Comb.*, 5:1–15, 2001.
2. M. Baroni, S. Grünwald, V. Moulton, and C. Semple. Bounding the number of hybridisation events for a consistent evolutionary history. *J. Math. Biol.*, 51:171–182, 2005.
3. R. G. Beiko and N. Hamilton. Phylogenetic identification of lateral genetic transfer events. *BMC Evol. Biol.*, 6:15, 2006.
4. M. L. Bonet, K. St. John, R. Mahindru, and N. Amenta. Approximating subtree distances between phylogenies. *J. Comp. Biol.*, 13:1419–1434, 2006.
5. M. Bordewich, C. McCartin, and C. Semple. A 3-approximation algorithm for the subtree distance between phylogenies. *J. Disc. Alg.*, 6:458–471, 2008.
6. M. Bordewich and C. Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Comb.*, 8:409–423, 2005.
7. M. Bordewich and C. Semple. Computing the hybridization number of two phylogenetic trees is fixed-parameter tractable. *IEEE/ACM Trans. on Comp. Biol. and Bioinf.*, 4:458–466, 2007.
8. M. Bordewich and C. Semple. Computing the minimum number of hybridization events for a consistent evolutionary history. *Disc. Appl. Math.*, 155:914–928, 2007.
9. F. Chataigner. Approximating the maximum agreement forest on  $k$  trees. *Inf. Proc. Letters*, 93:239–244, 2005.
10. M. Hallett and C. McCartin. A faster FPT algorithm for the maximum agreement forest problem. *Theory of Comp. Sys.*, 41:539–550, 2007.
11. J. Hein, T. Jiang, L. Wang, and K. Zhang. On the complexity of comparing evolutionary trees. *Disc. Appl. Math.*, 71:153–169, 1996.
12. G. Hickey, F. Dehne, A. Rau-Chaplin, and C. Blouin. SPR distance computation for unrooted trees. *Evol. Bioinf.*, 4:17–27, 2008.
13. D. M. Hillis, T. A. Heath, and K. St. John. Analysis and visualization of tree space. *Syst. Biol.*, 54:471–482, 2005.
14. D. M. Hillis, C. Moritz, and B. K. Mable, editors. *Molecular Systematics*. Sinauer Associates, 1996.
15. W. P. Maddison. Gene trees in species trees. *Syst. Biol.*, 46:523–536, 1997.
16. L. Nakhleh, T. Warnow, C. R. Lindner, and K. St. John. Reconstructing reticulate evolution in species—theory and practice. *J. Comp. Biol.*, 12:796–811, 2005.
17. E. M. Rodrigues, M.-F. Sagot, and Y. Wakabayashi. The maximum agreement forest problem: Approximation algorithms and computational experiments. *Theor. Comp. Sci.*, 374:91–110, 2007.
18. C. Whidden and N. Zeh. A unifying view on approximation and FPT of agreement forests. Technical Report CS-2009-02, Faculty of Computer Science, Dalhousie University, Halifax, Canada, 2009.