Linear-Time Algorithms for some Phylogenetic Tree Completion Problems under Robinson-Foulds Distance

Mukul S. Bansal

Department of Computer Science & Engineering and Institute for Systems Genomics,
University of Connecticut, Storrs, USA
mukul.bansal@uconn.edu

Abstract. We consider two fundamental computational problems that arise when comparing phylogenetic trees, rooted or unrooted, with non-identical leaf sets. The first problem arises when comparing two trees where the leaf set of one tree is a proper subset of the other. The second problem arises when the two trees to be compared have only partially overlapping leaf sets. The traditional approach to handling these problems is to first restrict the two trees to their common leaf set. An alternative approach that has shown promise is to first *complete* the trees by adding missing leaves, so that the resulting trees have identical leaf sets. This requires the computation of an optimal completion that minimizes the distance between the two resulting trees over all possible completions.
We provide optimal linear-time algorithms for both completion problems under the widely-used Robinson-Foulds (RF) distance measure. Our algorithm for the first problem improves the time complexity of the current fastest algorithm from quadratic (in the size of the two trees) to linear. No algorithms have yet been proposed for the more general second problem where both trees have missing leaves. We advance the study of this general problem by proposing a biologically meaningful restricted version of the general problem and providing optimal linear-time algorithms for the restricted version. Our experimental results on biological data sets suggest that using completion-based RF distances can result in different evolutionary inferences compared to traditional RF distances.

1 Introduction

A *phylogenetic tree*, or *phylogeny*, is a leaf-labeled tree that shows the evolutionary relationships between different biological entities, generally either species or genes. Phylogenies may be either rooted or unrooted. The leaf nodes of a phylogeny represent the extant set of entities on which the phylogeny is built, while internal nodes represent hypothetical ancestors. The comparison of different phylogenetic trees is one of the most fundamental tasks in evolutionary biology and computational phylogenetics. Many biologically relevant distance or similarity measures have been defined in the literature for the case when the two phylogenies to be compared have the same leaf set. These include the widely used Robinson-Foulds distance [27], triplet and quartet distances [13, 19], nearest neighbor interchange (NNI) and subtree prune and regraft (SPR) distances [20,30,33], maximum agreement subtrees [2,14,21], nodal distance [7], geodesic distance [23] and several others. Often, however, this comparison involves two

trees that have non-identical leaf sets. The need to compare trees that do not have identical leaf sets arises naturally in several situations: For instance, algorithms for computing phylogenetic supertrees are typically based on comparing input trees on partial leaf sets with candidate supertrees on the complete leaf set [1, 3, 9, 24, 31]. Likewise, searching for phylogenies similar to a query tree in a phylogenetic database [10, 25, 26, 29], and clustering of phylogenetic trees [34] often involve comparisons between trees with only partially overlapping leaf sets.

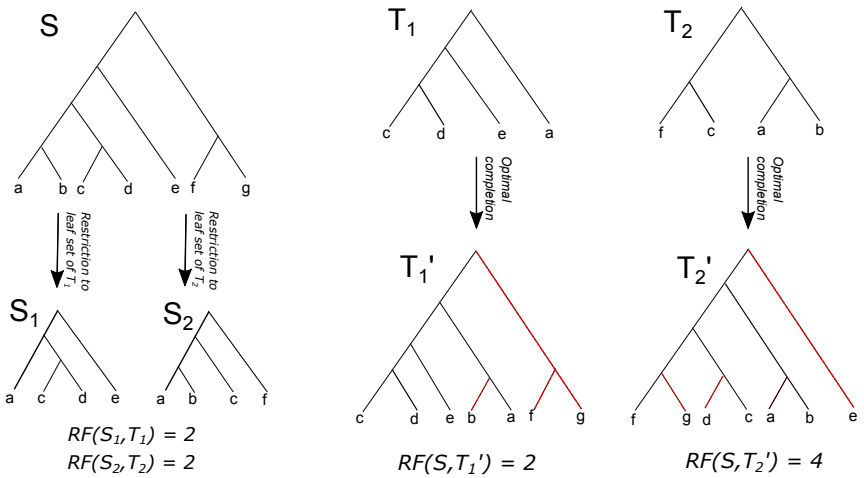

Fig. 1. RF(-) and RF(+) distances. This figure illustrates the difference between the traditional (RF(-)) and RF(+) distance measures when applied to trees with partially overlapping leaf sets. In this example, the leaf sets of T_1 and T_2 are a subset of the leaf set of S. To compute the RF(-) distance between T_1 and S, we must first restrict S to the leaf set of T_1, resulting in tree S_1. The RF(-) distance between S and T_1 is thus $RF(S_1, T_1)$, which is 2. Likewise, to compute the RF(-) distance between T_2 and S, we must first restrict S to the leaf set of T_2, resulting in tree S_2. The RF(-) distance between S and T_2 is thus $RF(S_2, T_2)$, which is also 2. In contrast, to compute the RF(+) distance between T_1 and S, we must first compute an optimal completion of T_1 on the leaf set of S (denoted by the dashed red lines), resulting in tree T_1'. The RF(+) distance between S and T_1 is thus $RF(S, T_1')$, which is 2. Likewise, to compute the RF(+) distance between T_2 and S, we must first compute an optimal completion of T_2 on the leaf set of S, resulting in tree T_2'. The RF(+) distance between S and T_2 is thus $RF(S, T_2')$, which is 4. Observe that while both T_1 and T_2 are equidistant from S under RF(-), computing the RF(+) distances reveals that T_1 is more similar to S than is T_2.

The traditional approach to comparing two phylogenies on non-identical leaf sets is to first restrict the two phylogenies to their common leaf set and then apply one of the distance or similarity measures that compare two trees on the same leaf set. However, an alternative, and perhaps more useful, approach to comparing trees with non-identical taxa is to *fill-in* or *complete* the two trees to be compared with the leaves missing from each, resulting in two trees on the same leaf set, and then apply the distance or similarity measure. This completion based approach is especially desirable when used with the

Robinson-Foulds (RF) distance measure [27], the most commonly used distance measure in evolutionary biology. Indeed, several important biological applications would directly benefit from the use of this completion-based RF distance, such as the construction of majority-rule(+) supertrees [12,17,18,22], construction of Robinson-Foulds supertrees [3,9,28], phylogenetic database search [10,25,26,29], and clustering of phylogenetic trees [34]. To distinguish between the two methods for computing RF distance between two trees with non-identical leaf sets, we refer to the completion-based RF distance as RF(+) distance and to the traditional pruning-based RF distance as RF(-). Figure 1 shows an example of two trees with partially overlapping leaf sets and these two ways of computing the RF distance between them.

Previous work. The idea of a completion-based RF(+) distance was proposed at least a decade ago. Cotton and Wilkinson were among the first to propose such a distance measure in their seminal paper describing majority-rule supertrees [12]. Specifically, they defined two types of majority-rule supertrees: majority-rule(-) and majority-rule(+) supertrees. The majority-rule(-) supertrees were based on traditional RF(-) distances between trees, while majority-rule(+) supertrees were based on completion-based RF(+) distances. Majority-rule(+) supertrees and its variants have been shown to have many desirable properties [16] and there have been efforts to develop exact (ILP based) and heuristic methods for computing majority-rule(+) supertrees [17, 22]. Though these methods only work for small datasets, they have been shown to result in biologically meaningful supertrees [17]. The paper by Kupczok [22] characterizes the RF(+) distance in the case when the leaf set of one tree is a subset of the leaf set of the other in terms of incompatible splits between the two trees, but does not provide an efficient algorithm for computing this distance or for computing an actual completion. More recently, Christensen et al. [11] provided an $O(n^2)$ time algorithm for the case when the leaf set of one tree is a subset of the leaf set of the other and applied the algorithm to compute optimal completions for gene trees with respect to a species tree. To the best of our knowledge, no algorithms (polynomial time or otherwise) currently exist for the general problem where the two trees have only partially overlapping leaf sets, or for any of its variants.

Our contribution. In this work, we address an important gap in the algorithmics of phylogenetic tree comparison. Specifically, we provide the first optimal, linear-time algorithms for two fundamental computational problems that arise when comparing phylogenetic trees with non-identical leaf sets. For the first problem, which arises when computing the RF(+) distance between two binary trees where the leaf set of one tree is a proper subset of the other, we improve upon the time complexity of the previous fastest algorithm for this problem by a factor of n, where n is the number of leaves in the larger of the two trees. For the second problem, which is a generalization of the first and arises when computing the RF(+) distance between two binary trees that have only partially overlapping leaf sets, we show that the default problem formulation can result in biologically meaningless results, propose a modification of the problem formulation that corrects this deficiency, and provide optimal linear-time algorithms for the modified problem. Crucially, no polynomial time algorithms currently exist for the default formulation of the second problem, and our modified problem formulation can be viewed as a biologically meaningful restricted version of the general problem. Our algorithms

are easy to understand and implement, work for both rooted and unrooted trees, and are scalable to the entire tree of life. These algorithms can be applied wherever phylogenetic distances must be computed between trees with non-identical leaf sets and enable new kinds of phylogenetic and comparative analyses that have been computationally infeasible.

We implemented our algorithm for the first problem and applied it to three published biological supertree data sets to study how RF(+) distances differ from RF(-) distances in practice. For each data set, we ordered the input trees according to their RF(+) and RF(-) distances to a precomputed supertree and measured how often the relative pairwise ranking between any pair of input trees differs between the two rankings. We found a large number of such pairs for each data set, demonstrating, for the first time, that using the RF(+) distance could result in different evolutionary inferences compared to inferences using the RF(-) distance.

RF(+) distances have several desirable properties compared to RF(-) distances. For instance, the range of possible values RF(+) distance can take ranges from 0 to about twice the size of the *union* of the leaf sets of the two trees, while for RF(-) distance this range is only from 0 to about twice the size of the *intersection* of the two leaf sets. Thus, RF(+) distances have significantly more discriminatory power than RF(-) distances. In applications such as median supertree construction, RF(+) distance has the distinct advantage that each input tree gets an equal "vote" in the supertree construction since all input trees contribute an RF distance within the same range. With RF(-) distances, larger trees can contribute much more to the total distance than smaller trees. Finally, in computing RF(-) distances we ignore the additional topological information provided by leaves that are present in only one tree, while RF(+) distance makes complete use of the information in the topologies of the two trees. RF(+) distances thus make more efficient use of the available information. Despite these advantages, RF(+) distances have not been applied in practice due to unavailability of efficient algorithms. In contrast, RF(-) distances can be computed in time linear in the sizes of the input trees. Our new algorithms address this discrepancy by making it equally computationally efficient to compute RF(+) distances.

The remainder of this manuscript is organized as follows. The next section includes basic definitions, notation, and problem formulations. Sections 3, 4, and 5 describe our algorithms for the problems considered in this work. Experimental results appear in Section 6 and concluding remarks appear in Section 7. For brevity, some proofs and certain details are deferred to the full version of this manuscript.

2 Preliminaries and Problem Definitions

Given a tree T, we denote its node set, edge set, and leaf set by $V(T)$, $E(T)$, and $\mathit{Le}(T)$, respectively. The set of all non-leaf (i.e., internal) nodes of T is denoted by $I(T)$.

If T is rooted, the root node of T is denoted by $\mathit{rt}(T)$, the parent of a node $v \in V(T)$ by $\mathit{pa}_T(v)$, its set of children by $\mathit{Ch}_T(v)$, and the (maximal) subtree of T rooted at v by $T(v)$. If two nodes in T have the same parent, they are called *siblings* of each other. The *least common ancestor*, denoted $\mathit{lca}_T(L)$, of a set $L \subseteq \mathit{Le}(T)$ in T is defined to be the node $v \in V(T)$ such that $L \subseteq \mathit{Le}(T(v))$ and $L \not\subseteq \mathit{Le}(T(u))$ for any child u of v.

A rooted tree is *binary* if all of its internal nodes have exactly two children, while an unrooted tree is *binary* if all its nodes have degree either 1 or 3. Throughout this work, the term *tree* refers to binary trees with uniquely labeled leaves.

Let T be a rooted or unrooted tree. Given a set $L \subseteq \mathit{Le}(T)$, let T' be the subtree of T with leaf set L. We define the *leaf induced subtree* $T[L]$ of T on leaf set L to be the tree obtained from T' by successively removing each non-root node of degree two and adjoining its two neighbors.

Definition 1 (Completion of a tree). *Given a tree T and a set L' such that $\mathit{Le}(T) \subseteq L'$, a* completion *of T on L' is a tree T' such that $\mathit{Le}(T') = L'$ and $T'[\mathit{Le}(T)] = T$.*

If T is a rooted tree, for each node $v \in V(T)$, the *clade* $C_T(v)$ is defined to be the set of all leaf nodes in $T(v)$; i.e. $C_T(v) = \mathit{Le}(T(v))$. We denote the set of all clades of a rooted tree T by $\mathit{Clade}(T)$. This concept can be extended to unrooted trees as follows. If T is an unrooted tree, each edge $(u, v) \in E(T)$ defines a partition of the leaf set of T into two disjoint subsets $\mathit{Le}(T_u)$ and $\mathit{Le}(T_v)$, where T_u is the subtree containing node u and T_v is the subtree containing node v, obtained when edge (u, v) is removed from T. The partition induced by any edge $(u, v) \in E(T)$ is called a *split* and is represented by the set $\{\mathit{Le}(T_u), \mathit{Le}(T_v)\}$. The set of all splits in an unrooted tree T is denoted by $\mathit{Split}(T)$.

The *symmetric difference* of two sets A and B, denoted by $A \Delta B$, is the set $(A \setminus B) \cup (B \setminus A)$.

Definition 2 (Robinson-Foulds distance). *The* Robinson-Foulds (RF) distance*, $RF(S, T)$, between two trees S and T is defined to be $|\mathit{Clade}(S) \Delta \mathit{Clade}(T)|$ if S and T are rooted trees, and $|\mathit{Split}(S) \Delta \mathit{Split}(T)|$ if S and T are unrooted trees.*

Let S and T be two trees. Without loss of generality, we will assume that $|\mathit{Le}(T)| \leq |\mathit{Le}(S)|$. When $\mathit{Le}(S) \neq \mathit{Le}(T)$, there are two possible scenarios: (1) $\mathit{Le}(T) \subsetneq \mathit{Le}(S)$, i.e., the leaf set of T is a proper subset of the leaf set of S, and (2) $\mathit{Le}(S) \cap \mathit{Le}(T) \subsetneq \mathit{Le}(T)$, i.e., each of S and T contains leaves not found in the other. Based on these two scenarios, and depending on whether the two trees are rooted or unrooted, we define the following four problems.

Problem 1 (Rooted One-Tree RF(+) (ROT-RF(+))) *Given two rooted trees S and T, such that $\mathit{Le}(T) \subseteq \mathit{Le}(S)$, compute a completion T' of T on $\mathit{Le}(S)$ such that $RF(S, T')$ is minimized.*

Problem 2 (Unrooted One-Tree RF(+) (UOT-RF(+))) *Given two unrooted trees S and T, such that $\mathit{Le}(T) \subseteq \mathit{Le}(S)$, compute a completion T' of T on $\mathit{Le}(S)$ such that $RF(S, T')$ is minimized.*

Problem 3 (Rooted RF(+) (R-RF(+))) *Given two rooted trees S and T, compute a completion S' of S on $\mathit{Le}(S) \cup \mathit{Le}(T)$ and a completion T' of T on $\mathit{Le}(S) \cup \mathit{Le}(T)$ such that $RF(S', T')$ is minimized.*

Problem 4 (Unrooted RF(+) (U-RF(+))) *Given two unrooted trees S and T, compute a completion S' of S on $\mathit{Le}(S) \cup \mathit{Le}(T)$ and a completion T' of T on $\mathit{Le}(S) \cup \mathit{Le}(T)$ such that $RF(S', T')$ is minimized.*

We show how to solve Problems 1 and 2 in $O(|V(S)|)$ time. As we will see later, Problems 3 and 4 can actually lead to biologically meaningless completions. We will therefore define biologically meaningful variants of Problems 3 and 4 (requiring only a slight variation on the original problems) and show how to solve them in $O(|V(S)| + |V(T)|)$ time. Throughout this work, we assume that the leaves of S and T are labeled by integers from the set $\{1, \ldots, |\mathit{Le}(S) \cup \mathit{Le}(T)|\}$. However, our algorithms work even if the leaf labels are arbitrary, and universal hashing [8] or perfect hashing [15] can be used to guarantee expected $O(|V(S)| + |V(T)|)$ time complexity.

3 A linear-time algorithm for ROT-RF(+)

To solve the ROT-RF(+) problem, our algorithm starts with the trees S and T and modifies T by adding to it, according to a particular scheme, the leaves from $\mathit{Le}(S) \setminus \mathit{Le}(T)$. The completed tree thus produced, denoted by T', will be such that $RF(S, T')$ is minimized.

We define *Tree-Add*(T, v, X) to be the tree obtained from T by attaching to it a tree X, where $\mathit{Le}(X) \cap \mathit{Le}(T) = \emptyset$, as follows: If v is not the root of T, then attach X onto the edge $(\mathit{pa}(v), v)$ (by subdividing $(\mathit{pa}(v), v)$ into two edges) such that $\mathit{rt}(X)$ becomes the sibling of the node $v \in V(T)$. If v is the root of T, then *Tree-Add*(T, v, X) is the tree obtained by creating a new root node and setting v and $\mathit{rt}(X)$ as its two children.

The main idea behind our algorithm can be illustrated by the following simple example. Suppose the given trees S and T are such that $\mathit{Le}(S) = \mathit{Le}(T) \cup \{l\}$. The goal is to add to T this leaf l, so as to minimize the RF distance. Let v denote the sibling of l in S. Let u denote the node $\mathit{lca}_T(\mathit{Le}(S(v)))$. As we will prove later, $T' = \mathit{Tree\text{-}Add}(T, u, l)$ must be an optimal completion for T. Our algorithm extends this idea to the case when T has multiple missing leaves. A description of the algorithm follows:

Algorithm *OneTreeCompletion*(S, T)

for each $v \in V(S)$ in post-order **do**
  Initialize the mapping $\mathcal{M}_S(v)$ to be NULL.
  if $v \in \mathit{Le}(S)$ **then**
    if leaf v is also present in tree T **then**
      Color v green.
    else
      Color v red.
  else
    if v has two green children **then**
      Color v green.
    else if v has two red children **then**
      Color v red.
    else if v has exactly one red child **then**
      Color v blue and label v as "marked".
    else
      Color v blue.
for each green or blue node v from $V(S)$ in post-order **do**
  Assign $\mathcal{M}_S(v) = \mathit{lca}_T(X)$, where $X = \{g | g \in \mathit{Le}(S(v))$ and g is green$\}$.

19: **for** each marked node $v \in V(S)$ in pre-order **do**
20: *Tree-Add*$(T, \mathcal{M}_S(v), R)$, where R is the subtree rooted at the red child of v.
21: Return the completed tree T.

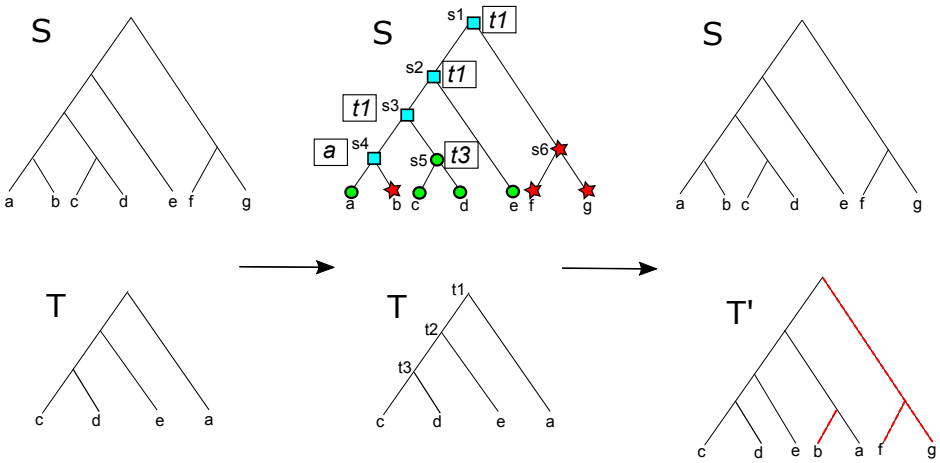

Fig. 2. Algorithm for ROT-RF(+). Given S and T as shown in the left column of the figure, Algorithm *OneTreeCompletion* first colors each node of S either green (circles), red (stars), or blue (squares) as shown in the middle column of the figure. A node is colored green if all leaves in the subtree rooted at that node are present in both S and T, red if all leaves in that subtree are present only in S, and blue if that subtree has both green and red descendants. If a blue node v has exactly one red child, then it is "marked". In this example, s_1 and s_4 are marked nodes. The algorithm then computes the LCA mapping, defined to be $lca_T(Le(S(v)) \cap Le(T))$, for each green or blue node v of S. These LCA mappings appear in the square boxes on S in the middle column. The algorithm then performs a pre-order traversal of S, grafting copies of the red subtrees at each marked node onto the appropriate edges of T. The grafted subtrees are shown using dashed red lines on T' in the right column. Tree T' is an optimal completion of T on $Le(S)$.

Figure 2 illustrates the algorithm through an example. Next, we prove the correctness and analyze the time complexity of this algorithm. We need the following additional definitions:

Definition 3 (Matched clade). *Given any two rooted trees A and B on the same leaf set, and $v \in V(A)$, we say that clade $C_A(v)$ has a* match *in B if Clade(B) contains $C_A(v)$.*

Definition 4 (Matchable clade of S). *Given any $v \in I(S)$, we call the clade $C_S(v)$* matchable *if there exists some completion of T on $Le(S)$ that contains the clade $C_S(v)$.*

The correctness of Algorithm *OneTreeCompletion* follows from the following lemma.

Lemma 1. *Let T' denote the completion of T returned by Algorithm OneTreeCompletion on trees S and T. Let T^* denote an optimal completion of T on $Le(S)$ that*

minimizes $RF(S, T^*)$. Then, $RF(S, T') = RF(S, T^*)$, implying that T' is a solution for the ROT-RF(+) problem.

Proof. It suffices to show that T' maximizes the number of matched clades $C_S(v)$, for $v \in V(S)$.

Observe that Algorithm *OneTreeCompletion* partitions $V(S)$ into three sets according to the color assigned to each node: red, green, or blue. We will consider these three sets of nodes separately.

Case 1: Red nodes. All maximal subtrees in S that contain only red nodes are included as-is in the completed tree T'. Thus, if v is a red node then $C_S(v)$ has a match in T'. Thus, T' maximizes the number of matched clades $C_S(v)$ over all red v.

Case 2: Green nodes. We claim that if v is green and $C_S(v)$ does not have a match in T' then it must be unmatchable. Suppose $C_S(v)$ has a match in T, and let $u \in V(T)$ be such that $C_S(v) = C_T(u)$. Observe that the clade $C_T(u)$ must also appear in T' since no blue node $x \in V(S)$ will be such that $\mathcal{M}_S(x) \in V(T(u))$. This implies that if $C_S(v)$ has a match in T then $C_S(v)$ must also have a match in T'. In other words, if $C_S(v)$ does not have a match in T' then $C_S(v)$ can not have a match in T. Now, since $C_S(v)$ only contains leaves that are already present in T, no completion of T on $\mathit{Le}(S)$ can create clade $C_S(v)$ if $C_S(v)$ is not already present in $\mathit{Clade}(T)$. Thus, if $C_S(v)$ has no match in T, then $C_S(v)$ must be unmatchable. This proves our claim, and so T' must maximize the number of matched clades $C_S(v)$ for green v.

Case 3: Blue nodes. We claim that if v is blue and $C_S(v)$ does not have a match in T' then it must be unmatchable. Let $C'_S(v)$ denote the set containing only the green nodes from $C_S(v)$. We will say that clade $C_S(v)$ has a partial-match in T if and only if $C'_S(v) \in \mathit{Clade}(T)$. Suppose $C_S(v)$ has a partial-match in T, and let u be the node from T for which $C_T(u) = C'_S(v)$ (note that, in fact, $u = \mathcal{M}_S(v)$). Observe that any marked node $x \in V(S(v))$ must be such that $\mathcal{M}_S(x) \in V(T(u))$. This implies that Algorithm *OneTreeCompletion* adds all the maximal red subtrees within $S(v)$ (i.e., subtrees rooted at a red child of a marked node in $S(v)$) to one or more of the edges in the set $\{(\mathit{pa}(t), t) | t \in T(u)\}$. Moreover, since $C_T(u) = C'_S(v)$, none of the other marked nodes $y \in V(S) \setminus V(S(v))$ can be such that $\mathcal{M}_S(y) \in V(T(u))$. Thus, there must be a node $u' \in T'$ for which $C_{T'}(u') = C_T(u) \cup \{r | r \text{ is a red leaf from } S(v)\}$, and so $C_S(v)$ must have a match in T'. Consequently, if $C_S(v)$ has a partial-match in T then $C_S(v)$ must have match in T'. In other words, if $C_S(v)$ does not have a match in T' then $C_S(v)$ can not have a partial-match in T.

Now, suppose $v \in V(S)$ is such that $C_S(v)$ has no partial-match in T. Since, $C'_S(v)$ only contains leaves that are already present in T, and there exists no node $u \in V(T)$ for which $C_T(u) = C'_S(v)$, no completion of T on $\mathit{Le}(S)$ can create clade $C_S(v)$. Thus, if $C_S(v)$ has no partial-match in T, then $C_S(v)$ must be unmatchable. This proves our claim, and so T' must maximize the number of matched clades $C_S(v)$ for blue v.

In summary, the tree T' maximizes the number of matched clades for each of the three sets into which $V(S)$ is partitioned, thereby maximizing the number of matched clades over all of $V(S)$. Hence, T' must be a solution for the ROT-RF(+) problem. □

Theorem 1. *Algorithm OneTreeCompletion solves the ROT-RF(+) problem in $O(|V(S)|)$ time.*

Proof. Lemma 1 establishes that Algorithm *OneTreeCompletion* solves the ROT-RF(+) problem. It therefore suffices to show that this algorithm can be implemented in $O(|V(S)|)$ time. We consider the complexity of each of the three 'for' loops separately.

The 'for' loop of Step 1 executes a single post-order traversal of the tree S, and so Steps 2 through 16 are executed a total of $O(|V(S)|)$ times. Each of the Steps 2 through 16, except for Step 16, clearly requires only $O(1)$ time per iteration. Step 16 can also be executed in $O(1)$ time after an $O(|S|)$ preprocessing step to construct a lookup table that enables $O(1)$ time lookup of whether a given leaf label from S occurs in tree T as well. This lookup table can be easily implemented using an array since the leaves of S (and T) are uniquely labeled by integers from the set $\{1, \ldots, |\mathit{Le}(S)|\}$. The indices of the array correspond to the leaf labels, and the entries correspond to whether the corresponding leaf appears only in S or in both T and S. Such an array can be constructed using a single traversal through the leaf sets of S and T. Even if the leaves have arbitrary labels, $O(|S|)$ preprocessing time and expected $O(1)$ lookup time can be achieved through hashing [8].

Step 18 is executed a total of $O(|V(S)|)$ times through the 'for' loop of Step 17. After an $O(|V(T)|)$ preprocessing step on T, the least common ancestor of any pair of nodes from $V(T)$ can be computed in constant time [5]. For any node v considered in the 'for' loop of Step 17, computing the least common ancestor mapping for that node (in Step 18) is equivalent to computing the least common ancestor of the mappings of its (up to two) blue or green children. Thus, after an $O(|\mathit{Le}(T)|)$ preprocessing step on T to enable fast least common ancestor computation [5], each execution of Step 18 requires only $O(1)$ time. This gives a total time complexity of $O(|V(S)|)$ for Steps 17 and 18.

The 'for' loop of Step 19 executes Step 20 a total of $O(|V(S)|)$ times. For a marked node v, Step 20 requires $O(|V(R)|)$ time, where R is the subtree rooted at the red child of v, to copy over the subtree R to T. Since each such R is disjoint from the others, over all possible marked nodes v, the total number of nodes in all the corresponding Rs is bounded by $O(|V(S)|)$. Thus, the total time complexity of Steps 19 and 20 is $O(|V(S)|)$.

Finally, Step 21 requires $O(|V(S)|)$ time to write the completed version of T. The total time complexity is thus $O(|V(S)|)$. □

Note that Algorithm *OneTreeCompletion* computes a single optimal completion, and that optimal completions need not be unique.

4 The R-RF(+) problem

Observe how an optimal completion of T in the ROT-RF(+) problem maximizes the number of clades that have a match in S. This ensures a biologically meaningful completion of T. However, in the R-RF(+) problem, where both trees may have missing leaves, it is possible that optimal completions of the two trees contain "extraneous" clades that contain leaves from both S and T but do not contain any leaves common to S and T. Extraneous clades are created by pairing a subtree containing only missing leaves from one tree with a subtree containing only missing leaves from the other tree.

Such clades can help to lower the RF distance between the two completed trees, but are not biologically meaningful since they are completely unsupported by the topologies of S and T. This phenomenon is illustrated through an example in Figure 3. We therefore define a biologically meaningful variant of the R-RF(+) problem that only allows completions that do not result in extraneous clades. Crucially, this restriction to only non-extraneous clades also makes the underlying completion problem easier to solve.

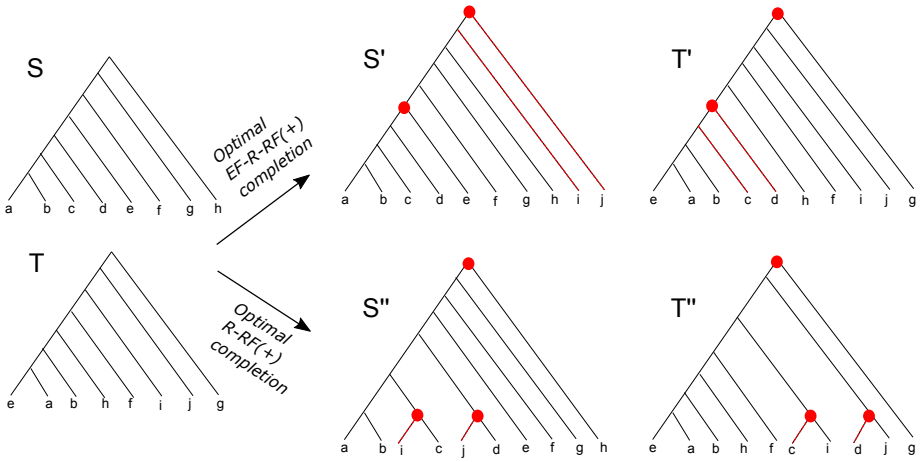

Fig. 3. Extraneous clades and R-RF(+) and EF-R-RF(+) completions. This figure shows two trees S and T with partial leaf set overlap whose optimal completions under the R-RF(+) problem result in extraneous clades. The tree S contains two leaves c and d that are absent from T, and the tree T contains two leaves i and j absent from S. The lower-right part of the figure shows optimal completions of S and T, labeled S'' and T'', respectively, that minimize the RF distance over all possible completions. The nodes marked in red denote (non-leaf) clades common to both S'' and T''. Observe that of the three nodes that S'' and T'' have in common, the lower two, i.e., $\{c, i\}$ and $\{d, j\}$ are extraneous clades that have no support in either S or T and do not contain any of the leaves shared by both S and T. Optimal completions under EF-R-RF(+) disallow such extraneous clades. The upper-right part of the figure shows optimal completions of S and T that minimize the RF distance over all completions without any extraneous clades. The completions S' and T' are more biologically meaningful since they only contain clades that have at least one leaf shared by both trees.

Definition 5 (Extraneous clade). *Suppose S and T are rooted trees. Given completions S' and T' of S and T, respectively, on $Le(S) \cup Le(T)$, we define a clade of S' or T' to be an* extraneous clade *if it contains leaves from both S and T but no leaves from $Le(S) \cap Le(T)$.*

Problem 5 (Extraneous-Clade-Free R-RF(+) (EF-R-RF(+))) *Given two rooted trees S and T, compute a completion S' of S on $Le(S) \cup Le(T)$ and a completion T' of T on $Le(S) \cup Le(T)$ such that S' and T' do not contain any extraneous clades and $RF(S', T')$ is minimized.*

An example of an optimal EF-R-RF(+) completion appears in Figure 3. Next, we show how to solve the EF-R-RF(+) problem in linear time.

4.1 A linear-time algorithm for EF-R-RF(+)

For the EF-R-RF(+) problem, $Le(S)$ and $Le(T)$ are both proper subsets of $Le(S) \cup Le(T)$, i.e., both S and T must be completed on the leaf set $Le(S) \cup Le(T)$. Our algorithm for this problem builds upon the algorithm for the ROT-RF(+) problem. Specifically, we first complete T on $Le(S) \cup Le(T)$ with respect to S, then complete S on $Le(S) \cup Le(T)$ with respect to the previous completion of T. Formally, the algorithm is as follows:

Algorithm *TwoTreeCompletion*(S, T)
1: $T' =$ *OneTreeCompletion*(S, T).
2: $S' =$ *OneTreeCompletion*(T', S).
3: **return** S' and T'.

In the following, we will show that when Algorithm *TwoTreeCompletion* terminates, the trees S' and T' returned by the algorithm must be such that they do not contain any extraneous clades, and that $RF(S', T')$ is the smallest possible for any completion of S and T that does not have extraneous clades. We will assume, without any loss of generality, that S and T have at least one leaf in common; if there are no leaves in common between S and T then the EF-R-RF(+) problem has no solution since any completion of S and T would necessarily contain extraneous clades.

For brevity, in the remainder of this section, we will implicitly assume that all completions of S and T are on the leaf set $Le(S) \cup Le(T)$. Next, we define the notions of *original nodes*, *grafted nodes*, and *grafted subtrees* in tree completions.

Definition 6 (Original nodes). *Let S' and T' denote any completions of S and T. Observe that completing a tree creates new internal nodes in the tree but preserves all original internal nodes (though not necessarily the clades rooted at those nodes). Thus, we have $I(S) \subset I(S')$ and $I(T) \subset I(T')$. The set of nodes in $I(S')$ that are also present in $I(S)$ are called the* original nodes *of S', denoted $\mathcal{O}(S')$. Analogously, the set of nodes in $I(T')$ that are also present in $I(T)$ are called the* original nodes *of T', denoted $\mathcal{O}(T')$.*

Definition 7 (Grafted nodes). *Let S' and T' denote any completions of S and T. Observe that any node $u \in I(S') \setminus \mathcal{O}(S')$ is either a node that was already present in a subtree from T (consisting of leaves missing from S) as that subtree was grafted into S, or a new node that was created as a subtree from T (consisting of leaves missing from S) was grafted into S. We refer to the new nodes created by the grafting of a subtree from T into S' as the* grafted nodes *of S', denoted $\mathcal{G}(S')$. Analogously, the set of nodes in $I(T') \setminus \mathcal{O}(T')$ that were newly created through the process of grafting a subtree from S into T are called the* grafted nodes *of T', denoted $\mathcal{G}(T')$.*

Definition 8 (Grafted subtrees). *If S' denotes any completion of S and $u \in \mathcal{G}(S')$, then u is created by the grafting of a subtree of T (consisting of leaves missing from S) at that node u in S'. We denote the grafted subtree of T at u by graft(u). Similarly, if*

T' denotes any completion of T and $v \in \mathcal{G}(T')$, then v is created by the grafting of a subtree of S at that node v in T'. We denote the grafted subtree of S at v by graft(v).

Node colorings. For convenience, we will color the nodes of S and T according to the coloring scheme used in Algorithm *OneTreeCompletion*. Thus, each node of S and T is colored either red, or green, or blue. We will assume that these colored nodes maintain their original colors in the completed trees S' and T', and thus both S' and T' contain nodes that are red, green, and blue, as well as nodes that are uncolored.

We now show that the completed trees S' and T' returned by Algorithm *TwoTreeCompletion* must be free of extraneous clades.

Lemma 2. *The trees S' and T' returned by Algorithm TwoTreeCompletion do not have any extraneous clades.*

Proof. Let us first consider the tree T'. Any non-original node in T' is either a node from a maximal red subtree of S or is a grafted node created by grafting a maximal red subtree of S into T' using the *Tree-Add* operation. Based on Algorithm *OneTreeCompletion*, each grafted node created through the *Tree-Add* operation has at least one green descendant, and so it cannot be extraneous. Moreover, any node inside a maximal red subtree of S only has descendants from S, not from T. Thus, since T did not contain any extraneous clades to begin with, neither can T'. An analogous argument applies to S'. □

The next lemma identifies an important property of optimal completions.

Lemma 3. *Let S^* and T^* be any optimal completions of S and T, respectively, under the EF-R-RF(+) problem. Then, for any $u \in \mathcal{G}(S^*)$, graft(u) must be a maximal red subtree of T and, for any $v \in \mathcal{G}(T^*)$, graft(v) must be a maximal red subtree of S.*

Proof. Observe that any maximal red subtree of T must appear as-is in the tree T^*, since grafting a red leaf or subtree from S into any of the red subtrees of T would result in an extraneous clade. We will show that if there exists a node $u \in \mathcal{G}(S^*)$ for which *graft*(u) is not a maximal red subtree of T, it is possible to modify the tree S^* so that the modified tree has more matched clades than S^*, a contradiction. An analogous argument applies to T^*. Suppose there exists such a node u. Then, there must exist a red internal node r of T such that the two subtrees, denoted R' and R'', rooted at the two children of r appear as-is in the tree S^* but not as siblings of each other (i.e., their roots do not have the same parent in S^*). Let r' and r'' denote the root nodes of R' and R'', respectively, and s' and s'' denote the parents of r' and r'' in S^*. Thus, $R' = \textit{graft}(s')$ and $R'' = \textit{graft}(s'')$. Now, observe that all clades of S^* rooted either at a node on the path from $\textit{lca}_{S^*}(s', s'')$ to s' or on the path from $\textit{lca}_{S^*}(s', s'')$ to s'', except for the node $\textit{lca}_{S^*}(s', s'')$ itself, must be mismatched clades (since all maximal red subtrees of T appear as-is in the tree T^*). Also, note that if S^* is modified by pruning out the subtree R' and regrafting it on the edge (s'', r''), then the only matched clades that can become mismatched are the ones whose roots lie on the path from $\textit{lca}_{S^*}(s', s'')$ to s' or from $\textit{lca}_{S^*}(s', s'')$ to s'', except for node $\textit{lca}_{S^*}(s', s'')$. Thus, modifying the tree S^* in this fashion does not result in any additional mismatched clades, but results in a new matched clade rooted at the node where R' is regrafted. Thus, the modified tree has a larger number of matched clades than S^*, which is a contradiction. □

We also have the following simple observation about optimal completions.

Observation 1 *Let S^* and T^* be optimal completions of S and T, respectively, that satisfy the property described in Lemma 3. Then any $u \in \mathcal{G}(S^*)$ and any $v \in \mathcal{G}(T^*)$ must have at least one green leaf as a descendant.*

Proof. This follows immediately from the fact that, under EF-R-RF(+), each clade must contain at least one green leaf (otherwise it would be an extraneous clade). □

Finally, the following lemma proves the correctness of Algorithm *TwoTreeCompletion*. For brevity, its proof is deferred to the full version of this paper.

Lemma 4. *Let S' and T' denote the completions of S and T, respectively, returned by Algorithm TwoTreeCompletion. Let S^* and T^* denote optimal completions of S and T, respectively, under the EF-R-RF(+) problem. Then, $RF(S', T') = RF(S^*, T^*)$.*

The next theorem now follows immediately based on Algorithm *TwoTreeCompletion*, Theorem 1, and Lemma 4.

Theorem 2. *Algorithm TwoTreeCompletion solves the EF-R-RF(+) problem in $O(|V(S)|+|V(T)|)$ time.*

5 Extension to unrooted trees

The linear-time algorithms for the ROT-RF(+) and EF-R-RF(+) problems described in the previous two sections can be easily extended to unrooted trees without any increase in time complexity. The idea is to first root the two unrooted trees at any leaf-edge that is common to both trees, and then apply the algorithm for ROT-RF(+) or EF-R-RF(+) on the resulting rooted trees. It can be shown that this is guaranteed to result in optimal solutions for UOT-RF(+) and EF-U-RF(+). Further details and proofs are deferred to the full version of this paper.

6 Experimental evaluation

We implemented our algorithm for the ROT-RF(+) problem and applied it to three large biological supertree data sets with the goal of assessing the impact of using RF(+) distance instead of the traditional RF(-) distance in practice. Specifically, we computed a supertree (using a standard supertree method; RFS [3] in this case) for each of the supertree data sets, and computed the RF(+) and RF(-) distances between the supertree and the input trees for each data set. Let the RF(+) distance between a supertree S and an input tree I be denoted by $RF^+(S, I)$, and the RF(-) distance those two trees by $RF^-(S, I)$. For each data set, we ordered the input trees according to their RF(+) and RF(-) distances to the supertree and measured how often the relative ranking between any pair of input trees differs between the two rankings. More precisely, given a supertree S and its set of input trees $\mathcal{I}$, we computed $RF^-(S, I)$ and $RF^+(S, I)$ for each

$I \in \mathcal{I}$, and counted the number of *Type-1*, *Type-2*, and *Type-3* pairs $\{I', I''\}$, where $I', I'' \in \mathcal{I}$, as follows:

Type-1 pairs. Pair $\{I', I''\}$ is Type-1 if either $RF^-(S, I') < RF^-(S, I'')$ but $RF^+(S, I') > RF^+(S, I'')$, or $RF^-(S, I') > RF^-(S, I'')$ but $RF^+(S, I') < RF^+(S, I'')$. These are pairs for which the RF(+) and RF(-) distances impose completely opposite orderings relative to the supertree.

Type-2 pairs. Pair $\{I', I''\}$ is Type-2 if $RF^-(S, I') = RF^-(S, I'')$ but $RF^+(S, I') \neq RF^+(S, I'')$. For these pairs, RF(-) distances are identical but RF(+) distances are not.

Type-3 pairs. Pair $\{I', I''\}$ is Type-3 if $RF^-(S, I') \neq RF^-(S, I'')$ but $RF^+(S, I') = RF^+(S, I'')$. For these pairs, RF(+) distances are identical but RF(-) distances are not.

The three data sets, marsupials [6], placental mammals [4], and legumes [32], contain 272, 116, and 571 species, and 158, 726, and 22 input trees, respectively. We observed that for the 158 input trees of the marsupial data set, there were 521 Type-1 pairs, 619 Type-2 pairs, and 376 Type-3 pairs. For the 726 input trees of the placental mammals data set, there were $5,816$ Type-1 pairs, $14,344$ Type-2 pairs, and $6,238$ Type-3 pairs. Likewise, for the 22 input trees in the legumes data set, we observed 8 Type-1 pairs, 3 Type-2 pairs, and no Type-3 pairs. These results show that there can be substantial difference between RF(-) and RF(+) distances and suggest that using RF(+) distances can result in different evolutionary inferences compared to inferences using RF(-).

Our current implementation is available from the author upon request. An improved open-source version, currently under development, will be released with the full version of this paper.

7 Conclusion

In this work, we provide the first optimal, linear-time algorithms for two fundamental computational problems that arise when comparing phylogenetic trees with non-identical leaf sets. For the first problem, which arises when computing the RF(+) distance between two trees where the leaf set of one tree is a proper subset of the other, we improved upon the time complexity of the previous fastest algorithm by a factor of n, where n is the size of the larger of the two trees. For the second problem, which arises when computing the RF(+) distance between two trees that have only partially overlapping leaf sets, and for which there are no existing algorithms, we defined a biologically meaningful restriction of the problem and provided an optimal linear-time algorithm for it. Our algorithms are easy to implement and should be scalable even to trees with millions of taxa. The algorithms work for both rooted and unrooted trees, and can be directly applied wherever phylogenetic distances must be computed between trees with non-identical leaf sets. Furthermore, our experiments with three large biological supertree data sets suggest that using the RF(+) distance can result in different evolutionary inferences compared to using the RF(-) distance.

The algorithms presented here have several important, well-established applications, including construction of majority-rule(+) supertrees and supertree construction in general, phylogenetic database search, and clustering of phylogenetic trees, and these applications should be studied and developed further. A more detailed experimental

study is needed to properly assess the impact of using RF(+) distances and to systematically study the effect of factors such as fraction of leaf set overlap and degree of discordance between trees. This work also motivates several theoretical questions for future investigation. For instance, our algorithms for the EF-R-RF(+) and EF-U-RF(+) problems cannot be easily extended to solve the R-RF(+) and U-RF(+) problems. In particular, if optimal completions are allowed to contain extraneous clades, then inferring the number and composition of these extraneous clades (to attain overall optimality) appears to be computationally challenging. It would be interesting to determine if linear or near-linear time algorithms exist for R-RF(+) and U-RF(+).

Funding: This work was supported in part by NSF awards IIS 1553421 and MCB 1616514 to MSB.

References

1. W. A. Akanni, M. Wilkinson, C. J. Creevey, P. G. Foster, and D. Pisani. Implementing and testing bayesian and maximum-likelihood supertree methods in phylogenetics. *Royal Society Open Science*, 2(8), 2015.
2. A. Amir and D. Keselman. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. *SIAM Journal on Computing*, 26(6):1656–1669, 1997.
3. M. S. Bansal, J. G. Burleigh, O. Eulenstein, and D. Fernández-Baca. Robinson-foulds supertrees. *Algorithms for Molecular Biology*, 5(1):18, Feb 2010.
4. R. Beck, O. Bininda-Emonds, M. Cardillo, F.-G. Liu, and A. Purvis. A higher-level MRP supertree of placental mammals. *BMC Evol. Biol.*, 6(1):93, 2006.
5. M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms*, 57(2):75–94, 2005.
6. M. Cardillo, O. R. P. Bininda-Emonds, E. Boakes, and A. Purvis. A species-level phylogenetic supertree of marsupials. *Journal of Zoology*, 264:11–31, 2004.
7. G. Cardona, M. Llabrés, F. Rosselló, and G. Valiente. Nodal distances for rooted phylogenetic trees. *Journal of Mathematical Biology*, 61(2):253–276, Aug 2010.
8. J. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.
9. R. Chaudhary, J. G. Burleigh, and D. Fernandez-Baca. Fast local search for unrooted robinson-foulds supertrees. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(4):1004–1013, 2012.
10. D. Chen, J. G. Burleigh, M. S. Bansal, and D. Fernández-Baca. Phylofinder: an intelligent search engine for phylogenetic tree databases. *BMC Evolutionary Biology*, 8(1):90, 2008.
11. S. Christensen, E. K. Molloy, P. Vachaspati, and T. Warnow. Optimal Completion of Incomplete Gene Trees in Polynomial Time Using OCTAL. In R. Schwartz and K. Reinert, editors, *17th International Workshop on Algorithms in Bioinformatics (WABI 2017)*, volume 88 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
12. J. A. Cotton, M. Wilkinson, and M. Steel. Majority-rule supertrees. *Systematic Biology*, 56(3):445–452, 2007.
13. D. E. Critchlow, D. K. Pearl, C. Qian, and D. Faith. The triples distance for rooted bifurcating phylogenetic trees. *Systematic Biology*, 45(3):323–334, 1996.
14. D. M. de Vienne, T. Giraud, and O. C. Martin. A congruence index for testing topological similarity between trees. *Bioinformatics*, 23(23):3119–3124, 2007.

15. M. Dietzfelbinger, A. Karlin, K. Mehlhorn, F. M. auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal on Computing*, 23(4):738–761, 1994.
16. J. Dong and D. Fernandez-Baca. Properties of majority-rule supertrees. *Systematic Biology*, 58(3):360–367, 2009.
17. J. Dong, D. Fernández-Baca, and F. McMorris. Constructing majority-rule supertrees. *Algorithms for Molecular Biology*, 5(1):2, Jan 2010.
18. J. Dong, D. Fernndez-Baca, F. McMorris, and R. C. Powers. An axiomatic study of majority-rule(+) and associated consensus functions on hierarchies. *Discrete Applied Mathematics*, 159(17):2038 – 2044, 2011.
19. G. F. Estabrook, F. R. McMorris, and C. A. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Zoology*, 34(2):193–200, 1985.
20. J. Felsenstein. *Inferring Phylogenies*. Sinauer Assoc., Sunderland, Mass, 2003.
21. C. R. Finden and A. D. Gordon. Obtaining common pruned trees. *Journal of Classification*, 2(1):255–276, Dec 1985.
22. A. Kupczok. Split-based computation of majority-rule supertrees. *BMC Evolutionary Biology*, 11(1):205, Jul 2011.
23. A. Kupczok, A. V. Haeseler, and S. Klaere. An exact algorithm for the geodesic distance between phylogenetic trees. *Journal of Computational Biology*, 15(6):577–591, 2008.
24. H. T. Lin, J. G. Burleigh, and O. Eulenstein. Triplet supertree heuristics for the tree of life. *BMC Bioinformatics*, 10(1):S8, Jan 2009.
25. M. M. McMahon, A. Deepak, D. Fernndez-Baca, D. Boss, and M. J. Sanderson. Stbase: One million species trees for comparative biology. *PLOS ONE*, 10(2):1–17, 02 2015.
26. W. H. Piel, M. Donoghue, M. Sanderson, and L. Netherlands. Treebase: a database of phylogenetic information. In *Proceedings of the 2nd International Workshop of Species 2000*, 2000.
27. D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131 – 147, 1981.
28. P. Vachaspati and T. Warnow. Fastrfs: fast and accurate robinson-foulds supertrees using constrained exact optimization. *Bioinformatics*, 33(5):631–639, 2017.
29. J. T. Wang, H. Shan, D. Shasha, and W. H. Piel. Fast structural search in phylogenetic databases. *Evolutionary Bioinformatics*, 2005(1):0–0, 2007.
30. M. Waterman and T. Smith. On the similarity of dendrograms. *Journal of Theoretical Biology*, 73(4):789 – 800, 1978.
31. C. Whidden, N. Zeh, and R. G. Beiko. Supertrees based on the subtree prune-and-regraft distance. *Systematic Biology*, 63(4):566–581, 2014.
32. M. Wojciechowski, M. Sanderson, K. Steele, and A. Liston. Molecular phylogeny of the "Temperate Herbaceous Tribes" of Papilionoid legumes: a supertree approach. In P. Herendeen and A. Bruneau, editors, *Advances in Legume Systematics*, volume 9, pages 277–298. Royal Botanic Gardens, Kew, 2000.
33. Y. Wu. A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics*, 25(2):190–196, 2009.
34. R. Yoshida, K. Fukumizu, and C. Vogiatzis. Multilocus phylogenetic analysis with gene tree clustering. *Annals of Operations Research*, Mar 2017.