Advancing Divide-And-Conquer Phylogeny Estimation Using Robinson-Foulds Supertrees

Xilin Yu[1]
Amazon AWS, Seattle, WA, USA
yuxilin51@gmail.com

Thien Le[2]
Department of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA.
thienle@mit.edu

Sarah Christensen
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
sac2@illinois.edu

Erin K. Molloy
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
emolloy2@illinois.edu

Tandy Warnow
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
warnow@illinois.edu

Abstract

One of the Grand Challenges in Science is the construction of the *Tree of Life*, an evolutionary tree containing several million species, spanning all life on earth. However, the construction of the Tree of Life is enormously computationally challenging, as all the current most accurate methods are either heuristics for **NP**-hard optimization problems or Bayesian MCMC methods that sample from tree space. One of the most promising approaches for improving scalability and accuracy for phylogeny estimation uses divide-and-conquer: a set of species is divided into overlapping subsets, trees are constructed on the subsets, and then merged together using a "supertree method". Here, we present Exact-RFS-2, the first polynomial-time algorithm to find an optimal supertree of two trees, using the Robinson-Foulds Supertree (RFS) criterion (a major approach in supertree estimation that is related to maximum likelihood supertrees), and we prove that finding the RFS of three input trees is **NP**-hard. We also present GreedyRFS (a greedy heuristic that operates by repeatedly using Exact-RFS-2 on pairs of trees, until all the trees are merged into a single supertree). We evaluate Exact-RFS-2 and GreedyRFS, and show that they have better accuracy than the current leading heuristic for RFS.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases supertrees, divide-and-conquer, phylogeny estimation

Digital Object Identifier 10.4230/LIPIcs.WABI.2020.15

Related Version A full version of the paper is available at `https://www.biorxiv.org/content/10.1101/2020.05.16.099895`.

Supplementary Material Exact-RFS-2 and GreedyRFS are available in open source form on Github at `https://github.com/yuxilin51/GreedyRFS`.

Funding *Sarah Christensen*: Ira and Debra Cohen Fellowship.
Erin K. Molloy: NSF Graduate Research Fellowship DGE-1144245 and the Ira and Debra Cohen Fellowship.
Tandy Warnow: US National Science Foundation grants 1458652 1513629, and 1535977.

[1] Corresponding author. This research was performed while Xilin Yu was a graduate student in the Department of Computer Science at the University of Illinois.
[2] This work was performed while Thien Le was an undergraduate at the University of Illinois

Acknowledgements This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

1 Introduction

Supertree construction (i.e., the combination of a collection of trees, each on a potentially different subset of the species, into a tree on the full set of species) is a natural algorithmic problem that has important applications to computational biology; see [7] for a 2004 book on the subject and [25, 38, 14, 18, 17, 16, 9, 10] for some of the recent papers on this subject. Supertree methods are particularly important for large-scale phylogeny estimation, where it can be used as a final step in a divide-and-conquer pipeline [45]: the species set is divided into two or more overlapping subsets, unrooted leaf-labelled trees are constructed (possibly recursively) on each subset, and then these subset trees are combined into a tree on the full dataset, using the selected supertree method. Furthermore, provided that optimal supertrees are computed, divide-and-conquer pipelines can be provably *statistically consistent* under stochastic models of evolution: i.e., as the amount of input data (e.g., sequence lengths when estimating gene trees, or number of gene trees when estimating species trees) increases, the probability that the true tree is returned converges to 1 [24, 44].

Unfortunately, the most accurate supertree methods are typically local-search heuristics for **NP**-hard optimization problems [4, 29, 25, 38, 33, 27, 34, 16], and are computationally intensive on large datasets. However, divide-and-conquer strategies, especially recursive ones, may only need to apply supertree methods to two trees at a time, and hence the computational complexity of supertree estimation given two trees is of interest. One optimization problem where optimal supertrees can be found on two trees is the **NP**-hard Maximum Agreement Supertree (SMAST) problem (also known as the Agreement Supertree Taxon Removal problem), which removes a minimum number of leaves so that the reduced trees have an agreement supertree [17, 14]. Similarly, the Maximum Compatible Supertree (SMCT) problem, which removes a minimum number of leaves so that the reduced trees have a compatibility supertree [5, 6], can also be solved in polynomial time on two trees (and note that SMAST and SMCT are identical when the input trees are fully resolved). Because SMAST and SMCT remove taxa, methods for these optimization problems are not *true supertree methods*, because they do not return a tree on the entire set of taxa. However, solutions to SMAST and SMCT could potentially be used as *constraints* for other supertree methods, where the deleted leaves are added into the computed SMAST or SMCT trees, so as to optimize the desired criterion.

When restricting to methods that return trees on the full set of taxa, much less seems to be understood about finding supertrees on two trees. However, if the two input trees are compatible (i.e., there is a supertree that equals or refines each tree when restricted to the respective leaf set), then finding that compatibility supertree is solvable in polynomial time, using (for example) the well known BUILD algorithm [1], but more efficient algorithms exist (e.g., [6, 3]).

Since compatibility is a strong requirement (rarely seen in biological datasets), optimization problems are more relevant. One optimization problem worth discussing is the Maximum Agreement Supertree Edge Contraction problem (which takes as input a set of rooted trees and seeks a minimum number of edges to collapse so that an agreement supertree exists). This problem is **NP**-hard, but the decision problem can be solved in $O((2k)^p kn^2)$ time when

the input has k trees and p is the allowed number of number of edges to be collapsed [14]. Note that the algorithm for AST-EC proposed by [14] may be exponential even for two trees, when the number of edges that must be collapsed is $\Omega(n)$ (e.g., imagine two caterpillar trees, where one is obtained from the other by moving the left-most leaf to the rightmost position).

In sum, while supertree methods are important and well studied, when restricted to the major optimization problems that do not remove taxa, polynomial time methods do not seem to be available, even for the special case where the input contains just two trees. This restriction has consequences for large-scale phylogeny estimation, as without good supertree methods, divide-and-conquer pipelines are not guaranteed to be statistically consistent, are not fast, and do not have good scalability [44].

In this paper we examine the well known Robinson-Foulds Supertree (RFS) problem [2], which seeks a supertree that minimizes the total Robinson-Foulds [30] distance to the input trees. Although RFS is **NP**-hard [20], it has several desirable properties: it is closely related to maximum likelihood supertrees [36] and, as shown very recently, has good theoretical performance for species tree estimation in the presence of gene duplication and loss [23]. Because of its importance, there are several methods for RFS supertrees, including PluMiST [18], MulRF [8], and FastRFS [42]. A comparison between FastRFS and other supertree methods (MRL [25], ASTRAL, ASTRID [41], PluMiST, and MulRF) on simulated and biological supertree datasets showed that FastRFS matched or improved on the other methods with respect to topological accuracy and RFS criterion scores [42]. Hence, FastRFS is currently the leading method for the RFS optimization problem.

The main contributions of this paper are:

- We prove that RFS is solvable in $O(n^2|X|)$ time for two trees, where n is the number of leaves and X is the number of shared leaves (Theorem 2) and **NP**-hard for three or more trees (Lemma 10).
- We present Exact-RFS-2, a polynomial time algorithm for the RFS problem when given only two source trees, and explore its performance on simulated data, both within a natural divide-and-conquer pipeline and within a greedy heuristic (Section 3). We show that Exact-RFS-2 outperforms FastRFS [42] on two trees, the current most accurate method for RFS, and that GreedyRFS is better than FastRFS for small to moderate numbers of source trees (Section 4).
- We prove that divide-and-conquer pipelines using Exact-RFS-2 are statistically consistent methods for phylogenetic tree estimation (both gene trees and species trees) under standard sequence evolution models (Theorem 12).
- We establish equivalence between RFS and some other supertree problems (Lemma 1).
- We show critical differences between RFS and SMAST/SMCT problems, that establish that methods for SMAST or SMCT cannot provably be used to constrain the search for RFS supertrees (Lemma 23 in Appendix in the full version on bioRxiv).

The remainder of the paper is organized as follows. In Section 2, we provide terminology and define the optimization problems we consider. We present the Exact-RFS-2 algorithm and establish theory related to the algorithm in Section 3. Our experimental performance study is presented in Section 4, and we conclude in Section 5 with a discussion of trends and future research directions.

2 Terminology and Problem Statements

We let $[N] = \{1, 2, \ldots, N\}$ and $\mathcal{A} = \{T_i \mid i \in [N]\}$ denote the input to a supertree problem, where each T_i is a phylogenetic tree on leaf set $L(T_i) = S_i \subseteq S$ (where $L(t)$ denotes the leaf set of t) and the output is a tree T where $L(T)$ is the set of all species that appear as a leaf

in at least one tree in $\mathcal{A}$, which we will assume is all of S. We use the standard supertree terminology, and refer to the trees in $\mathcal{A}$ as "source trees" and the set $\mathcal{A}$ as a "profile". For a tree T, let $V(T)$ and $E(T)$ denote the set of vertices and edges of T, respectively.

Robinson-Foulds Supertree

Each edge e in a tree T defines a bipartition $\pi_e := [A|B]$ of the leaf set, and each tree is defined by the set $C(T) := \{\pi_e \mid e \in E(T)\}$. The *Robinson-Foulds distance* [30] (also called the bipartition distance) between trees T and T' with the same leaf set is $\mathrm{RF}(T, T') := |C(T) \backslash C(T')| + |C(T') \backslash C(T)|$. We extend the definition of RF distance to allow for T and T' to have different leaf sets as follows: $RF(T, T') := RF(T|_X, T'|_X)$, where X is the shared leaf set and $t|_X$ denotes the homeomorphic subtree of t induced by X. Letting $\mathcal{T}_S$ denote the set of all phylogenetic trees such that $L(T) = S$ and $\mathcal{T}_S^B$ denote the binary trees in $\mathcal{T}_S$, then a Robinson-Foulds supertree [2] of a profile $\mathcal{A}$ is a binary tree

$$T_{\mathrm{RFS}} = \operatorname*{argmin}_{T \in \mathcal{T}_S^B} \sum_{i \in [N]} \mathrm{RF}(T, T_i).$$

We let $\mathrm{RF}(T, \mathcal{A}) := \sum_{i \in [N]} \mathrm{RF}(T, T_i)$ denote the *RFS score* of T with respect to profile $\mathcal{A}$. Thus, the **Robinson-Foulds Supertree problem** takes as input the profile $\mathcal{A}$ and seeks a Robinson-Foulds (RF) supertree for $\mathcal{A}$, which we denote by $\mathrm{RFS}(\mathcal{A})$.

Split Fit Supertree

The Split Fit (SF) Supertree problem was introduced in [46], and is based on optimizing the number of shared splits (i.e., bipartitions) between the supertree and the source trees. For two trees T, T' with the same leaf set, the *split support* is the number of shared bipartitions, i.e., $\mathrm{SF}(T, T') := |C(T) \cap C(T')|$. For trees with different leaf sets, we restrict them to the shared leaf set before calculating the split support. The Split Fit supertree for a profile $\mathcal{A}$ of source trees, denoted $\mathrm{SFS}(\mathcal{A})$, is a tree $T_{\mathrm{SFS}} \in \mathcal{T}_S^B$ such that

$$T_{\mathrm{SFS}} = \operatorname*{argmax}_{T \in \mathcal{T}_S^B} \sum_{i \in [N]} \mathrm{SF}(T, T_i).$$

Thus, the split support score of T with respect to $\mathcal{A}$ is $\mathrm{SF}(T, \mathcal{A}) := \sum_{i \in [N]} \mathrm{SF}(T, T_i)$. The **Split Fit Supertree (SFS) problem** takes as input the profile $\mathcal{A}$ and seeks a Split Fit supertree (the supertree with the maximum split support score), which we denote by $\mathrm{SFS}(\mathcal{A})$.

Nomenclature for variants of RFS and SFS problems

- The relaxed versions of the problems where we do not require the output to be binary (i.e., we allow $T \in \mathcal{T}_S$) are named Relax-RFS and Relax-SFS.
- We append "-N" to the name to indicate that we assume there are N source trees. If no number is specified then the number of source trees is unconstrained.
- We append "-B" to the name to indicate that the source trees are required to be binary; hence, we indicate that the source trees are allowed to be non-binary by not appending -B.

Thus, the RFS problem with two binary input trees is RFS-2-B and the relaxed SFS problem with three (not necessarily binary) input trees is Relax-SFS-3.

Other notation

For any $v \in V(T)$, we let $N_T(v)$ denote the set of neighbors of v in T. A tree T' is a *refinement* of T iff T can be obtained from T' by contracting a set of edges. Two bipartitions π_1 and π_2 of the same leaf set are said to be *compatible* if and only if there exists a tree T such that $\pi_i \in C(T), i = 1, 2$. A bipartition $\pi = [A|B]$ restricted to a subset R is $\pi|_R = [A \cap R|B \cap R]$. For a graph G and a set F of vertices or edges, we use $G + F$ to represent the graph obtained from adding the set F of vertices or edges to G, and $G - F$ is defined for deletions, similarly.

3 Theoretical Results

In this section we establish the main theoretical results, with detailed proofs provided in [47] or in the Appendix in the full version on bioRxiv [48].

3.1 Solving RFS and SFS on two trees

▶ **Lemma 1.** *Given an input set $\mathcal{A}$ of source trees, a tree $T \in \mathcal{T}_S^B$ is an optimal solution for RFS($\mathcal{A}$) if and only if it is an optimal solution for SFS($\mathcal{A}$).*

The main result of this paper is Theorem 2 (correctness is proved later within the main body of the paper, and the running time is established in the Appendix):

▶ **Theorem 2.** *Let $\mathcal{A} = \{T_1, T_2\}$ with S_i the leaf set of T_i ($i = 1, 2$) and $X := S_1 \cap S_2$. The problems RFS-2-B($\mathcal{A}$) and SFS-2-B($\mathcal{A}$) can be solved in $O(n^2|X|)$ time, where $n := \max\{|S_1|, |S_2|\}$.*

3.1.1 Exact-RFS-2: Polynomial time algorithm for RFS-2-B and SFS-2-B

The input to Exact-RFS-2 is a pair of binary trees T_1 and T_2. Let X denote the set of shared leaves. At a high level, Exact-RFS-2 constructs a tree T_{init} that has a central node that is adjacent to every leaf in X and to the root of every "rooted extra subtree" (a term we define below under "Additional notation") so that T_{init} contains all the leaves in S. It then modifies T_{init} by repeatedly refining it to add specific desired bipartitions, to produce an optimal Split Fit (and optimal Robinson-Foulds) supertree (Figure 3). The bipartitions that are added are defined by a maximum independent set in a bipartite "weighted incompatibility graph" we compute.

Additional notation

Let 2^X denote the set of all bipartitions of X; any bipartition that splits a single leaf from the remaining $|X| - 1$ leaves will be called "trivial" and the others will be called "non-trivial". Let $C(T_1, T_2, X)$ denote $C(T_1|_X) \cup C(T_2|_X)$, and let Triv and NonTriv denote the sets of trivial and non-trivial bipartitions in $C(T_1, T_2, X)$, respectively. We refer to $T_i|_X, i = 1, 2$ as **backbone trees** (Figure 2). Recall that we suppress degree-two vertices when restricting a tree T_i to a subset X of the leaves; hence, every edge e in $T_i|_X$ will correspond to an edge or a path in T (see Fig. 1 for an example). We will let $P(e)$ denote the path associated to edge e, and let $w(e) := |P(e)|$ (the number of edges in $P(e)$). Finally, for $\pi \in C(T_i|_X)$, we define $e_i(\pi)$ to be the edge that induces π in $T_i|_X$ (Fig. 1).

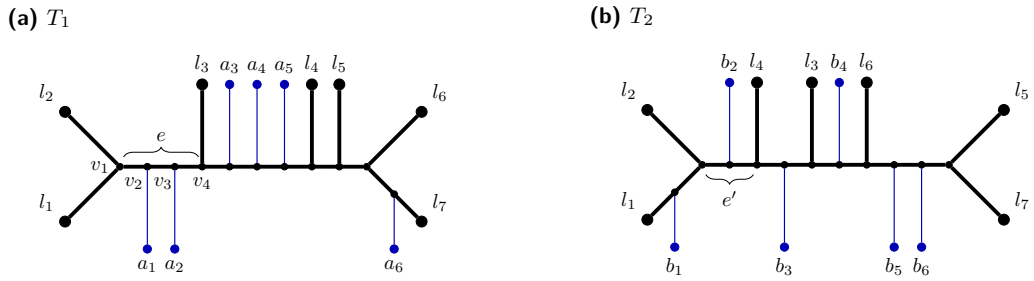

Figure 1 T_1 and T_2 depicted in (a) and (b) have an overlapping leaf set $X = \{l_1, l_2, \ldots, l_7\}$. Each of $a_1, \ldots, a_6$ and $b_1, \ldots, b_6$ can represent a multi-leaf extra subtree. For $e \in T_1|_X$ as shown, $P(e)$ is the path from v_1 to v_4, so $w(e) = 3$. Using indices to represent the shared leaves, let $\pi = [12|34567]$; then $e_1(\pi) = e$ and $e_2(\pi) = e'$. $\mathcal{TR}(e) = \{a_1, a_2\}$, $\mathcal{TR}(e') = \{b_2\}$, so $\mathcal{TR}^*(\pi) = \{a_1, a_2, b_2\}$. Let $A = \{1, 2, 3\}$, $B = \{4, 5, 6, 7\}$. Ignoring the trivial bipartitions, we have $\mathcal{BP}(A) = \{[12|34567]\}$ and $\mathcal{BP}(B) = \{[1234|567], [12345|67], [12346|57]\}$. $\mathcal{TRS}(A) = \{a_1, a_2, b_1, b_2\}$ and $\mathcal{TRS}(B) = \{a_6, b_4, b_5, b_6\}$.

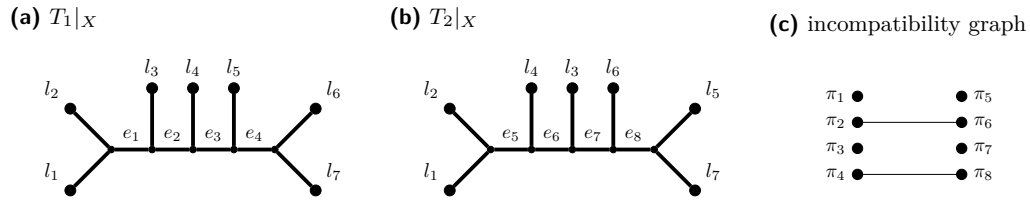

Figure 2 We show (a) $T_1|_X$, (b) $T_2|_X$, and (c) their incompatibility graph, based on the trees T_1 and T_2 in Figure 1 (without the trivial bipartitions). Each π_i is the bipartition induced by e_i, and the weights for $\pi_1, \ldots, \pi_8$ are $3, 4, 1, 1, 2, 2, 2, 3$, in that order. We note that π_1 and π_5 are the same bipartition, but they have different weights as they are induced by different edges; similarly for π_3 and π_7. The maximum weight independent set in this graph has all the isolated vertices $(\pi_1, \pi_3, \pi_5, \pi_7)$ and also π_2, π_8, and so has total weight 15.

The next concept we introduce is the set of **extra subtrees**, which are rooted subtrees of T_1 and T_2, formed by deleting X and all the edges and vertices on paths between vertices in X (i.e., we delete $T_i|_X$ from T_i). Each component in $T_i - T_i|_X$ is called an **extra subtree** of T_i, and note that the extra subtree t is naturally seen as rooted at the unique vertex $r(t)$ that is adjacent to a vertex in $T_i|_X$. Thus, $\text{Extra}(T_i) = \{t \mid t \text{ is a component in } T_i - T_i|_X\}$.

We can now define the initial tree T_{init} computed by Exact-RFS-2: T_{init} has a center node that is adjacent to every $x \in X$ and also to the root $r(t)$ for every extra subtree $t \in \text{Extra}(T_1) \cup \text{Extra}(T_2)$. Note that T_{init} has a leaf for every element in S, and that $T_{\text{init}}|_{S_i}$ is a contraction of T_i, formed by collapsing all the edges in the backbone tree $T_i|_X$.

We say that an extra subtree t is **attached to edge** $e \in E(T_i|_X)$ if the root of t is adjacent to an internal node of $P(e)$, and we let $\mathcal{TR}(e)$ denote the set of such extra subtrees attached to edge e. Similarly, if $\pi \in C(T_1, T_2, X)$, we let $\mathcal{TR}^*(\pi)$ refer to the set of extra subtrees that attach to edges in a backbone tree that induce π in either $T_1|_X$ or $T_2|_X$. For example, if both trees T_1 and T_2 contribute extra subtrees to π, then $\mathcal{TR}^*(\pi) := \bigcup_{i \in [2]} \mathcal{TR}(e_i(\pi))$.

For any $Q \subseteq X$, we let $\mathcal{BP}_i(Q)$ denote the set of bipartitions in $C(T_i|_X)$ that have one side being a strict subset of Q, and we let $\mathcal{TRS}_i(Q)$ denote the set of extra subtrees associated with these bipartitions. In other words, $\mathcal{BP}_i(Q) := \{[A|B] \in C(T_i|_X) \mid A \subsetneq Q \text{ or } B \subsetneq Q\}$, and $\mathcal{TRS}_i(Q) := \bigcup_{\pi \in \mathcal{BP}_i(Q)} \mathcal{TR}(e_i(\pi))$. Intuitively, $\mathcal{TRS}_i(Q)$ denotes the set of extra subtrees in T_i that are "on the side of Q". By Corollary 14, which appears in the Appendix, for any $\pi =$

Algorithm 1 Exact-RFS-2: Computing a Robinson-Foulds supertree of two trees (see Figure 3).

Input: two binary trees T_1, T_2 with leaf sets S_1 and S_2 where $S_1 \cap S_2 = X \neq \emptyset$
Output: a binary supertree T on leaf set $S = S_1 \cup S_2$ that maximizes the split support score
1: compute $C(T_1|_X)$ and $C(T_2|_X)$
2: **for** each $\pi = [A|B] \in C(T_1, T_2, X)$ **do**
3: **for** $i \in [2]$ **do**
4: compute $\mathcal{TR}(e_i(\pi))$, $w(e_i(\pi))$
5: compute $\mathcal{BP}(A)$, $\mathcal{BP}(B)$, $\mathcal{TRS}(A)$, $\mathcal{TRS}(B)$, and $\mathcal{TR}^*(\pi)$,
6: construct T as a star tree with leaf set X and center vertex $\hat{v}$ and with the root of each $t \in \text{Extra}(T_1) \cup \text{Extra}(T_2)$ connected to $\hat{v}$ by an edge ▷ let $T_{\text{init}} = T$
7: construct the weighted incompatibility graph G of $T_1|_X$ and $T_2|_X$
8: compute the maximum weight independent set I^* in G
9: let I be the set of bipartitions associated with vertices in I^*
10: **for** each $\pi = [\{a\}|B] \in \text{Triv}$ **do**
11: detach all extra subtrees in $\mathcal{TR}^*(\pi)$ from $\hat{v}$ and attach them onto $(\hat{v}, a)$ such that $\mathcal{TR}(e_1(\pi))$ are attached first with their ordering matching their attachments on $e_1(\pi)$ and $\mathcal{TR}(e_2(\pi))$ are attached to the right of all subtrees in $\mathcal{TR}(e_1(\pi))$ with the ordering of them also matching their attachments on $e_2(\pi)$
▷ let $\tilde{T} = T$ after for loop
12: $H(\hat{v}) = \text{NonTriv}$, set $sv(\pi) = \hat{v}$ for all $\pi \in \text{NonTriv}$
13: **for** each $\pi \in \text{NonTriv} \cap I$ (in any order) **do**
14: $T \leftarrow \text{Refine}(T, \pi, H, sv)$ ▷ let $T^* = T$ after for loop
15: arbitrarily refine T to make it a binary tree
16: return T

$[A|B] \in C(T_i|_X)$, $\mathcal{BP}_i(A) \cup \mathcal{BP}_i(B)$ is the set of bipartitions in $C(T_i|_X)$ that are compatible with π. Finally, let $\mathcal{BP}(Q) = \mathcal{BP}_1(Q) \cup \mathcal{BP}_2(Q)$, and $\mathcal{TRS}(Q) = \mathcal{TRS}_1(Q) \cup \mathcal{TRS}_2(Q)$. We give an example for these terms in Figure 1.

The *incompatibility graph* of a set of trees, each on the same set of leaves, has one vertex for each bipartition in any tree (and note that bipartitions can appear more than once) and edges between bipartitions if they are incompatible (see [28]). We compute a **weighted incompatibility graph** for the pair of trees $T_1|_X$ and $T_2|_X$, in which the weight of the vertex corresponding to bipartition π appearing in tree $T_i|_X$ is $w(e_i(\pi))$, as defined previously. Thus, if a bipartition is common to the two trees, it produces two vertices in the weighted incompatibility graph, and each vertex has its own weight (Figure 2).

We divide $\mathcal{C} = C(T_1) \cup C(T_2)$ into two sets: $\Pi_1 = \{[A|B] \in \mathcal{C} \mid A \cap X \neq \emptyset \text{ and } B \cap X \neq \emptyset\}$, and $\Pi_2 = \{[A|B] \in \mathcal{C} \mid A \cap X = \emptyset \text{ or } B \cap X = \emptyset\}$. Intuitively, Π_1 is the set of bipartitions from the input trees that are induced by edges in the minimal subtree of T_1 or T_2 spanning X, and Π_2 are all the other input tree bipartitions. We define $p_1(\cdot)$ and $p_2(\cdot)$ on trees $T \in \mathcal{T}_S$ by:

$$p_1(T) = \sum_{i \in [2]} |C(T|_{S_i}) \cap C(T_i) \cap \Pi_1|, \quad p_2(T) = \sum_{i \in [2]} |C(T|_{S_i}) \cap C(T_i) \cap \Pi_2|.$$

Note that $p_1(T)$ and $p_2(T)$ decompose the split support score of T into the score contributed by bipartitions in Π_1 and the score contributed by bipartitions in Π_2; thus, the split support score of T with respect to T_1, T_2 is $p_1(T) + p_2(T)$.

As we will show, the two scores can be maximized independently and we can use this observation to refine T_{init} so that it achieves the optimal total score.

Overview of Exact-RFS-2

Exact-RFS-2 (Algorithm 1) has four phases. In the pre-processing phase (lines 1–5), it computes the weight function w and the mappings $\mathcal{TR}, \mathcal{TR}^*, \mathcal{BP}$, and $\mathcal{TRS}$ for use in latter parts of Algorithm 1 and Algorithm 2. In the initial construction phase (line 6), it constructs

a tree T_{init} (as described earlier), and we note that T_{init} maximizes $p_2(\cdot)$ score (Lemma 3). In the refinement phase (lines 7–14), it refines T_{init} so that it attains the maximum $p_1(\cdot)$ score. In the last phase (line 15), it arbitrarily refines T to make it binary. The refinement phase begins with the construction of a weighted incompatibility graph G of $T_1|_X$ and $T_2|_X$ (see Figure 2). It then finds a maximum weight independent set of G that defines a set $I \subseteq C(T_1, T_2, X)$ of compatible bipartitions of X. Finally, it uses these bipartitions of X in I to refine T_{init} to achieve the optimal $p_1(\cdot)$ score, by repeatedly applying Algorithm 2 for each $\pi \in I$ (and we note that the order does not matter). See Figure 3 for an example of Exact-RFS-2 given two input source trees.

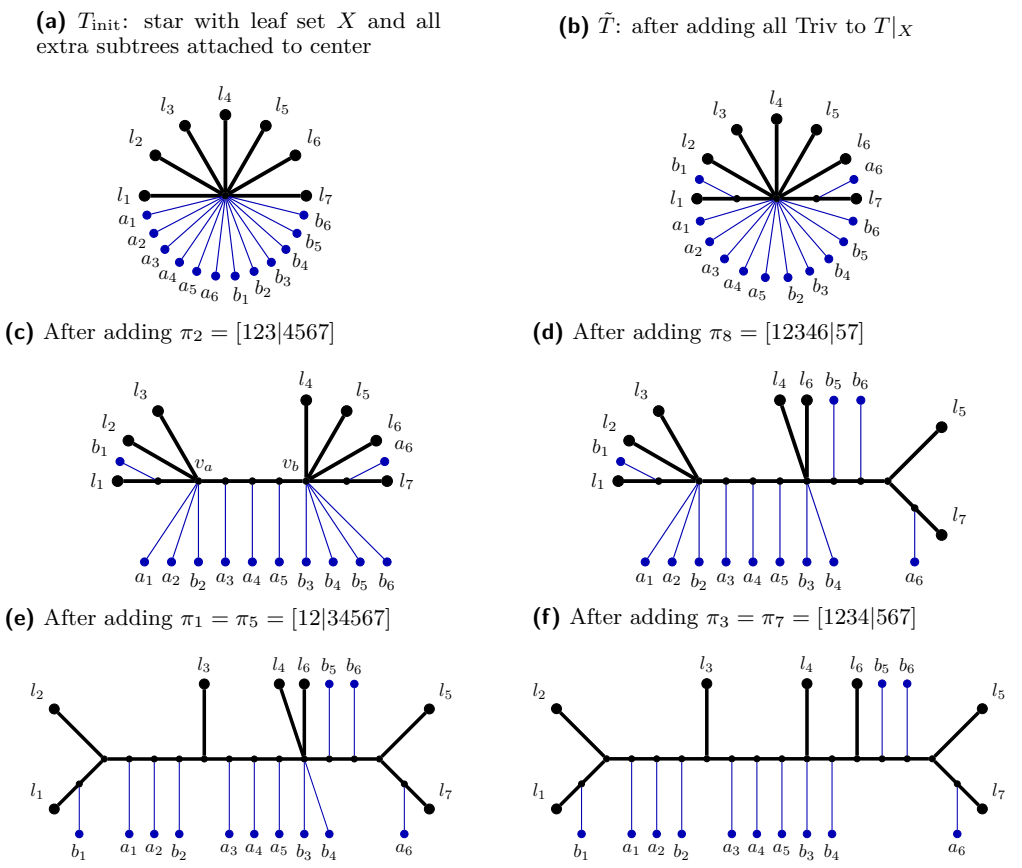

Figure 3 Algorithm 1 working on T_1 and T_2 from Figure 1 as source trees; the indices of leaves in $X = \{l_1, l_2, \ldots, l_7\}$ represent the leaves and the notation of $\pi_1, \ldots, \pi_8$ is from Figure 2. In (a) to (f), the $p_X(\cdot)$ score of the trees are $14, 16, 20, 23, 27, 29$, in that order. We explain how the algorithm obtain the tree in (c) from $\tilde{T}$ by adding $\pi_2 = [123|4567]$ to the backbone of $\tilde{T}$. Let $A = \{l_1, l_2, l_3\}$ and $B = \{l_4, l_5, l_6, l_7\}$. The center vertex c of $\tilde{T}$ is split into two vertices v_a, v_b with an edge between them. Then all neighbors of c between c and A are made adjacent to v_a while the neighbors between c and B are made adjacent to v_b. All neighbors of c which are roots of extra subtrees are moved around such that all extra subtrees in $\mathcal{TR}^*(\pi_2)$ are attached onto (v_a, v_b); all extra subtrees in $\mathcal{TRS}(A) = \{a_1, a_2, b_2\}$ are attached to v_a and all extra subtrees in $\mathcal{TRS}(B) = \{b_4, b_5, b_6\}$ are attached to v_b. We note that in this step, b_3 can attach to either v_a or v_b because it is not in $\mathcal{TRS}(A)$ or $\mathcal{TRS}(B)$. However, when obtaining the tree in (d) from the tree in (c), b_3 can only attach to the left side because for $A' = \{l_1, l_2, l_3, l_4, l_6\}$, $[124|3567] \in \mathcal{BP}(A')$ and thus $b_3 \in \mathcal{TRS}(A')$.

Algorithm 2 refines the given tree T on leaf set S with bipartitions on X from $C(T_1, T_2, X) \setminus C(T|_X)$. Given bipartition $\pi = [A|B]$ on X, Algorithm 2 produces a refinement T' of T such that $C(T'|_{S_i}) = C(T|_{S_i}) \cup \{\pi' \in C(T_i) \mid \pi'|_X = \pi\}$ for both $i = 1, 2$. To do this, we first

Algorithm 2 Refine.

Input: a tree T on leaf set S, a nontrivial bipartition $\pi = [A|B]$ of X, two data structures H and sv
Output: a tree T' which is a refinement of T such that for both $i = 1, 2$, $C(T'|_{S_i}) = C(T|_{S_i}) \cup \{\pi' \in C(T_i) \mid \pi'|_X = \pi\}$
1: $v \leftarrow sv(\pi)$
2: $T' \leftarrow T + v_a + v_b + (v_a, v_b)$
3: compute $N_A := \{u \in N_T(v) \mid \exists a \in A$ s.t. u can reach a in $T - v\}$ and $N_B := \{u \in N_T(v) \mid \exists b \in B$ s.t. u can reach b in $T - v\}$.
4: **for** each $u \in N_A \cup N_B$ **do**
5: **if** $u \in N_A$ **then** connect u to v_a
6: **else** connect u to v_b
7: detach all extra subtrees in $\mathcal{TR}^*(\pi)$ from v and attach them onto (v_a, v_b) such that $\mathcal{TR}(e_1(\pi))$ are attached first with their ordering matching their attachments on $e_1(\pi)$ and $\mathcal{TR}(e_2(\pi))$ are attached to the right of all subtrees in $\mathcal{TR}(e_1(\pi))$ with the ordering of them also matching their attachments on $e_2(\pi)$
8: **for** each $t \in \mathcal{TRS}(A)$ **do**
9: if t is attached to v, detach it and attach to v_a
10: **for** each $t \in \mathcal{TRS}(B)$ **do**
11: if t is attached to v, detach it and attach to v_b
12: **for** each remaining extra subtree attached to v **do**
13: detach it from v and attach it to either v_a or v_b
14: $H(v_a) \leftarrow \emptyset, H(v_b) \leftarrow \emptyset$
15: **for** each $\pi' \in H(v)$ **do**
16: **if** $\pi' \in \mathcal{BP}(A)$ **then**
17: $sv(\pi') = v_a$, $H(v_a) \leftarrow H(v_a) \cup \{\pi'\}$
18: **else if** $\pi' \in \mathcal{BP}(B)$ **then**
19: $sv(\pi') = v_b$, $H(v_b) \leftarrow H(v_b) \cup \{\pi'\}$
20: **else**
21: discard π'
22: return $T' = T' - v$

find the unique vertex v such that no component of $T - v$ has leaves from both A and B. We create two new vertices v_a and v_b with an edge between them. We divide the neighbor set of v into three sets: N_A is the set of neighbors that split v from leaves in A, N_B is the set of neighbors that split v from leaves in B, and N_{other} contains the remaining neighbors. Then, we make vertices in N_A adjacent to v_a and vertices in N_B adjacent to v_b. We note that $N_{\text{other}} = \emptyset$ if $X = S$ and thus there are no extra subtrees. In the case where $X \neq S$, N_{other} contains the roots of the extra subtrees adjacent to v and we handle them in four different cases to make T' include the desired bipartitions:

- those vertices that root extra subtrees in $\mathcal{TR}^*(\pi)$ are moved onto the edge (v_a, v_b) (by subdividing the edge to create new vertices, and then making these vertices adjacent to the new vertices)
- those vertices that root extra subtrees in $\mathcal{TRS}(A)$ are made adjacent to v_a
- those that root extra subtrees in $\mathcal{TRS}(B)$ are made adjacent to v_b
- the remaining vertices can be made adjacent to either v_a or v_b

Algorithms 1 and 2 also use two data structures (functions) H and sv: (1) For a given node $v \in V(T)$, $H(v) \subseteq C(T_1, T_2, X)$ is the set of bipartitions of X that can be added to $T|_X$ by refining $T|_X$ at v, and (2) Given $\pi \in C(T_1, T_2, X)$, $sv(\pi) = v$ means $\exists T'$, a refinement of T at v, so that $C(T'|_X) = C(T|_X) \cup \{\pi\}$.

▶ **Lemma 3.** *For any tree $T \in \mathcal{T}_S$, $p_2(T) \leq |\Pi_2|$. In particular, let T_{init} be the tree defined in line 6 of Algorithm 1. Then, $p_2(T_{\text{init}}) = |\Pi_2|$.*

Lemma 3 formally states that the tree T_{init} we build in line 6 of Exact-RFS-2 (Algorithm 1) maximizes the $p_2(\cdot)$ score. This lemma is true because there are only $|\Pi_2|$ bipartitions that can contribute to $p_2(\cdot)$ and T_{init} contains all of them by construction. We define the function $w^* : \Pi \to \mathbb{N}_{\geq 0}$ as follows:

$$w^*(\pi) = \begin{cases} 0 & \text{if } \pi \notin C(T_1, T_2, X), \\ w(e_1(\pi)) & \text{if } \pi \in C(T_1|_X) \setminus C(T_2|_X), \\ w(e_2(\pi)) & \text{if } \pi \in C(T_2|_X) \setminus C(T_1|_X), \\ \sum_{i\in[2]} w(e_i(\pi)) & \text{else.} \end{cases}$$

For any set F of bipartitions, we let $w^*(F) = \sum_{\pi\in F} w^*(\pi)$.

▶ **Lemma 4.** *Let $\pi = [A|B] \in \Pi$. Let $T \in \mathcal{T}_S$ be any tree with leaf set S such that $\pi \notin C(T|_X)$ but π is compatible with $C(T|_X)$. Let T' be a refinement of T such that for all $\pi' \in C(T'|_{S_i}) \setminus C(T|_{S_i})$ for some $i \in [2]$, $\pi'|_X = \pi$. Then, $p_1(T') - p_1(T) \leq w^*(\pi)$.*

▶ **Lemma 5.** *For any compatible set $F \subseteq \Pi$, let $T \in \mathcal{T}_S$ be any tree with leaf set S such that $C(T|_X) = F$. Then $p_1(T) \leq w^*(F)$.*

Lemma 4 shows that $w^*(\pi)$ represents the maximum potential increase in $p_1(\cdot)$ as a result of adding bipartition π to $T|_X$. The proof of Lemma 4 follows the idea that for any bipartition π of X, there are at most $w^*(\pi)$ edges in either T_1 or T_2 whose induced bipartitions become π when restricted to X. Therefore, by only adding π to $T|_X$, at most $w^*(\pi)$ more bipartitions get included in $C(T|_{S_1})$ or $C(T|_{S_2})$ so that they contribute to the increase of $p_1(T)$. The proof of Lemma 5 uses Lemma 4 repeatedly by adding the compatible bipartitions to the tree in an arbitrary order.

▶ **Proposition 6.** *Let $\tilde{T}$ be the tree constructed after line 11 of Algorithm 1, then $p_1(\tilde{T}) = w^*(\text{Triv})$.*

The proof naturally follows by construction (Line 8 of Algorithm 1), and implies that the algorithm adds the trivial bipartitions of X (which are all in I) to $T|_X$ so that $p_1(T)$ reaches the full potential of adding those trivial bipartitions.

▶ **Lemma 7.** *Let T be a supertree computed within Algorithm 1 at line 14 immediately before a refinement step. Let $\pi = [A|B] \in \text{NonTriv} \cap I$. Let T' be a refinement of T obtained from running Algorithm 2 with supertree T, bipartition π, and the auxiliary data structures H and sv. Then, $p_1(T') - p_1(T) = w^*(\pi)$.*

The idea for the proof of Lemma 7 is that for any non-trivial bipartition $\pi \in I$ of X to be added to $T|_X$, Algorithm 2 is able to split the vertex correctly and move extra subtrees around in a way such that each bipartition in T_1 or T_2 that is induced by an edge in $P(e_1(\pi))$ or $P(e_2(\pi))$, which is not in $T|_{S_1}$ or $T|_{S_2}$ before the refinement, becomes present in $T|_{S_1}$ or $T|_{S_2}$ after the refinement. Since there are exactly $w^*(\pi)$ such bipartitions, they increase $p_1(\cdot)$ by $w^*(\pi)$.

▶ **Proposition 8.** *Let G be the weighted incompatibility graph on $T_1|_X$ and $T_2|_X$, and let I be the set of bipartitions associated with vertices in I^*, which is a maximum weight independent set of G. Let F be any compatible subset of $C(T_1, T_2, X)$. Then $w^*(I) \geq w^*(F)$.*

We now restate and prove Theorem 2:

▶ **Theorem 2.** *Let $\mathcal{A} = \{T_1, T_2\}$ with S_i the leaf set of T_i ($i = 1, 2$) and $X := S_1 \cap S_2$. The problems RFS-2-B($\mathcal{A}$) and SFS-2-B($\mathcal{A}$) can be solved in $O(n^2|X|)$ time, where $n := \max\{|S_1|, |S_2|\}$.*

Proof. First we claim that $p_1(T^*) \geq p_1(T)$ for any tree $T \in \mathcal{T}_S$, where T^* is defined as from line 14 of Algorithm 1. Fix arbitrary $T \in \mathcal{T}_S$ and let $F = C(T|_X)$. Then by Lemma 5, $p_1(T) \leq w^*(F)$. We know that $w^*(\pi) = 0$ for any $\pi \notin C(T_1, T_2, X)$, so $w^*(F) = w^*(F \cap C(T_1, T_2, X))$ and thus $p_1(T) \leq w^*(F \cap C(T_1, T_2, X))$. Since $F \cap C(T_1, T_2, X)$ is a compatible subset of $C(T_1, T_2, X)$, we have $w^*(F \cap C(T_1, T_2, X)) \leq w^*(I)$ by Proposition 8. Then $p_1(T) \leq w^*(I)$. Since $\text{Triv} \subseteq C(T_1|_X) \cap C(T_2|_X) \subseteq I$, we have $I = (\text{NonTriv} \cap I) \cup (\text{Triv} \cap I) = (\text{NonTriv} \cap I) \cup \text{Triv}$. Therefore, by Proposition 6 and Lemma 7, we have

$$p_1(T^*) = p_1(\tilde{T}) + \sum_{\pi \in \text{NonTriv} \cap I} w^*(\pi) = w^*(\text{Triv}) + w^*(\text{NonTriv} \cap I) = w^*(I).$$

Therefore, $p_1(T^*) = w^*(I) \geq p_1(T)$.

From Lemma 3 and the fact that a refinement of a tree never decreases $p_1(\cdot)$ and $p_2(\cdot)$, we also know that $p_2(T^*) \geq p_2(T_{\text{init}}) \geq p_2(T)$ for any tree $T \in \mathcal{T}_S$. Since for any $T \in \mathcal{T}_S$, $\text{SF}(T, \mathcal{A}) = p_1(T) + p_2(T)$, T^* achieves the maximum split support score with respect to $\mathcal{A}$ among all trees in $\mathcal{T}_S$. Thus, T^* is a solution to Relax–SFS-2-B (Corollary 9). If T^* is not binary, Algorithm 1 arbitrarily resolves every high degree node in T^* until it is a binary tree and then returns a tree that achieves the maximum split support score among all binary trees of leaf set S. See the Appendix for the running time analysis. ◀

▶ **Corollary 9.** *Let $\mathcal{A} = \{T_1, T_2\}$ with S_i the leaf set of T_i ($i = 1, 2$) and $X := S_1 \cap S_2$. Relax–SFS-2-B can be solved in $O(n^2|X|)$ time, where $n := \max\{|S_1|, |S_2|\}$.*

▶ **Lemma 10.** *RFS-3, SFS-3, and Relax–SFS-3 are all* **NP***-hard.*

The proof for this lemma can be found in the Appendix.

3.2 DACTAL-Exact-RFS-2

Let Φ be a model of evolution (e.g., GTR) for which statistically consistent methods exist, and we have some data (e.g., sequences) generated by the model and wish to construct a tree. We construct an initial estimate of the tree, and we select an edge e in the tree. The deletion of e and its endpoints creates four subtrees, and we let P be the set of the p nearest leaves to e taken from each subtree (including all leaves that tie for closest in each subtree). We define the subsets be $A \cup P$ and $B \cup P$, where $\pi_e = [A|B]$), and we re-estimate trees on these subsets and then combine the trees together using Exact-RFS-2. We call this the DACTAL-Exact-RFS-2 pipeline, due to its similarity to the DACTAL pipeline [24]. The DACTAL pipeline differs from the DACTAL-Exact-RFS-2 pipeline only in that it computes four trees (each containing the set P and otherwise being leaf-disjoint) and then combines the overlapping subset trees using the Strict Consensus Merger technique, and was proven statistically consistent when the subset trees are computed using statistically consistent methods.

Before we prove that DACTAL-Exact-RFS-2 can enable statistically consistent pipelines, we begin with some definitions. Given a tree T and an internal edge e in T, the deletion of the edge e and its endpoints defines four subtrees. A **short quartet around** e is a set of four leaves, one from each subtree, selected to be closest to the edge. Note that due to ties, there can be multiple short quartets around some edges. The set of short quartets for a tree T is the set of all short quartets around the edges of T. The **short quartet trees of** T is the set of quartet trees on short quartets induced by T. It is well known that the short quartet trees of a tree T define T, and furthermore T can be computed from this set in polynomial time [11, 12, 13].

▶ **Lemma 11.** *Let T be a binary tree on leaf set S and let $A, B \subseteq S$. Let $T_A = T|_A$ and $T_B = T|_B$ (i.e., T_A and T_B are induced subtrees). If every short quartet tree is induced in T_A or in T_B, then T is the unique compatibility supertree for T_A and T_B and Exact-2-RFS(T_A, T_B) $= T$.*

Proof. Because T_A and T_B are induced subtrees of T, it follows that T is a compatibility supertree for T_A and T_B. Furthermore, because every short quartet tree appears in at least one of these trees, T is the unique compatibility supertree for T_A and T_B (by results from [12, 13], mentioned above). Finally, because T is a compatibility supertree, the RFS score of T with respect to T_A, T_B is 0, which is the best possible. Since Exact-2-RFS solves the RFS problem on two binary trees, Exact-2-RFS returns T given input T_A and T_B. ◀

Thus, Exact-2-RFS is guaranteed to return the true tree when given two correct trees that have sufficient overlap (in that all short quartets are included). We continue with proving that these pipelines are statistically consistent.

▶ **Theorem 12.** *The DACTAL-Exact-RFS-2 pipeline is a statistically consistent method for estimating the unrooted tree topology under any model Φ for which statistically consistent unrooted tree topology estimation methods exist.*

Proof. The proof is very similar to the proof given for the original DACTAL pipeline in [24]. Let Φ be the stochastic evolution model. To establish statistical consistency of the DACTAL-Exact-RFS-2 pipeline (see above), we need to prove that as the amount of data increases the unrooted tree topology that is returned by the pipeline converge to the true unrooted tree topology. That is, we will show that for any $\epsilon > 0$, there is an amount of data so that the probability of returning the true tree topology given that amount of data is at least $1 - \epsilon$. Hence, let F be the method used to compute the starting tree, let G be the method used to compute the subset trees, and let $\epsilon > 0$ be given. Because F is statistically consistent under Φ, it follows that there is an amount of data so that the starting tree computed by F will have the true tree topology T with probability at least $1 - \epsilon/2$. Now consider the decomposition into two sets produced by the algorithm produced by deleting edge e, applied to a tree with the true unrooted tree topology. Note that for any $p \geq 1$, all the leaves appearing in any short quartet around e are placed in the set P. Now, subset trees are computed using G on $A \cup P$ and $B \cup P$, where $\pi_e = [A|B]$, which we will refer to as T_A and T_B, respectively. Since G is statistically consistent, for a large enough amount of data, T_A and T_B will have the true tree topology on their leaf sets ($T|_{L(T_A)}$ and $T|_{L(T_B)}$, respectively) with probability at least $1 - \epsilon/2$. When T_A and T_B are equal to the true trees on their leaf sets, then every short quartet tree of T is in T_A or T_B, so that by Lemma 11, T is the only compatibility supertree for T_A and T_B. Thus, under these conditions, Exact-2-RFS(T_A, T_B) returns T. Hence, for a large enough amount of data, Exact-2-RFS(T_A, T_B) returns T with probability at least $1 - \epsilon$, completing our proof. ◀

Hence, DACTAL+Exact-2-RFS is statistically consistent under all standard molecular sequence evolution models and also under the MSC+GTR model [43, 31] where gene trees evolve within species trees under the multi-species coalescent model (which addresses gene tree discordance due to incomplete lineage sorting [19]) and then sequences evolve down each gene tree under the GTR model.

Note that all that is needed for F and G to guarantee that the pipeline is statistically consistent is that they should be statistically consistent under Φ. However, for the sake of improving empirical performance, F should be fast so that it can run on the full dataset but G can be more freely chosen, since it will only be run on smaller datasets. Indeed, the user

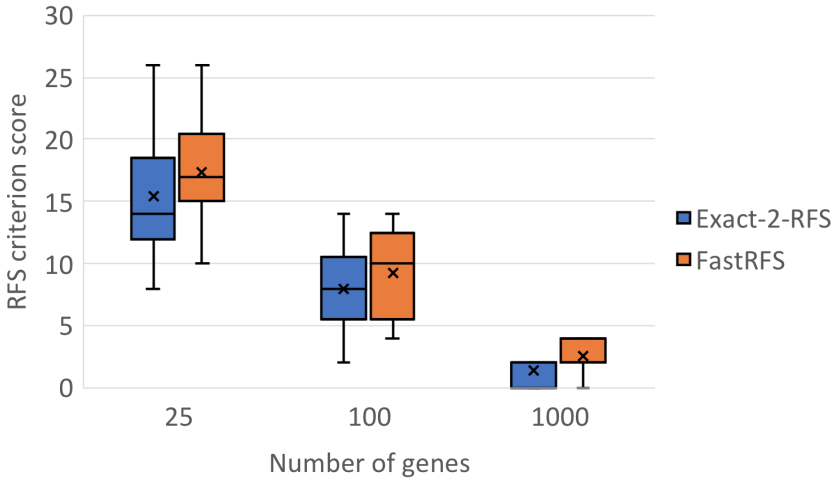

Figure 4 Results for Experiment 1: Exact-2-RFS has better RFS criterion scores than FastRFS (lower is better) in ILS-based species tree estimation (using ASTRAL-III [49], for 501 species with varying numbers of genes).

can specify the size of the subsets that are analyzed, with smaller datasets enabling the use of more computationally intensive methods.

For example, when estimating trees under the GTR [40] model, F could be a fast but statistically consistent distance-based method such as neighbor joining [32] and G could be RAxML [35], a leading maximum likelihood method. For the MSC+GTR model, F and G could be polynomial time summary methods (i.e., methods that estimate the species tree by combining gene trees), with F being ASTRID [41] (a very fast summary method) and G being ASTRAL [21, 22, 49], which is slower than ASTRID but often more accurate. However, if the subsets are chosen to be very small, then other choices for G include StarBeast2 [26], a Bayesian method for co-estimating gene trees and species trees under the MSC+GTR model.

4 Experiments and Results

We performed two experiments: Experiment 1, where we used Exact-2-RFS within a divide-and-conquer strategy for large scale phylogenomic species tree estimation where gene trees differ from the species tree due to Incomplete Lineage Sorting (ILS), and Experiment 2, where we used Exact-2-RFS as part of a greedy heuristic, GreedyRFS, for large-scale supertree estimation.

4.1 Experiment 1: Phylogenomic species tree estimation

In this experiment, the input is a set of gene trees that can differ from the species tree due to Incomplete Lineage Sorting [19], ASTRAL [21, 22, 49] is used to construct species trees on the two overlapping subsets in the DACTAL pipeline described above, and the two overlapping estimated species trees are then merged together using either Exact-2-RFS or FastRFS. Because the divide-and-conquer strategy produces two source trees, the RFS criterion score for Exact-2-RFS cannot be worse than the score obtained by FastRFS; here we examine the degree of improvement. The simulation protocol produced datasets with high variability (especially for small numbers of genes), so that there was substantial range in the optimal criterion scores for 25 and 100 genes (Fig. 4).

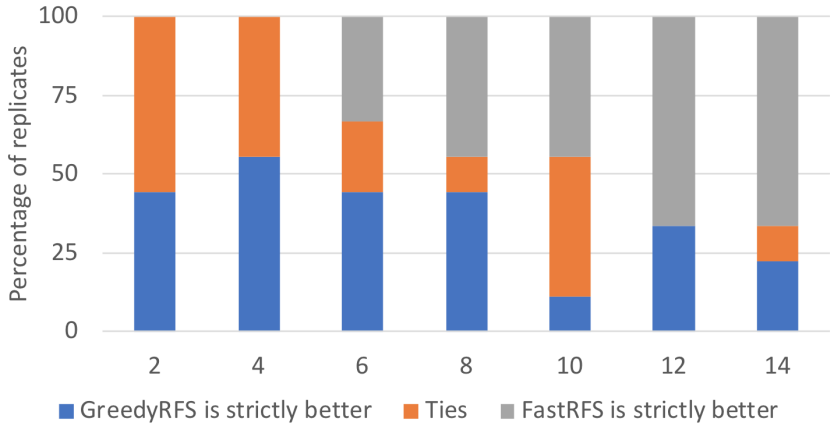

Figure 5 Results for Experiment 2: The percentage of datasets (y-axis) that each method (FastRFS and GreedyRFS) ties with or is strictly better than the other in terms of RFS criterion score is shown for varying numbers of source trees (x-axis), based on nine replicate supertree 500-leaf 20% scaffold datasets (from [39]).

On average, Exact-2-RFS produces better RFS scores than FastRFS for all numbers of genes (Fig. 4), showing that divide-and-conquer pipelines are improved using Exact-2-RFS compared to FastRFS.

4.2 Experiment 2: Exploring GreedyRFS for supertree estimation

We developed GreedyRFS, a greedy heuristic that takes a profile $\mathcal{A}$ as input, and then merges pairs of trees until all the trees are merged into a single tree. The choice of which pair to merge follows the technique used in SuperFine [39] for computing the Strict Consensus Merger, which selects the pair that maximizes the number of shared taxa between the two trees (other techniques could be used, potentially with better accuracy [15]). Thus, GreedyRFS is identical to Exact-2-RFS when the profile has only two trees.

We use a subset of the SMIDgen [37] datasets with 500 species and varying numbers of source trees (each estimated using maximum likelihood heuristics) that have been used to evaluate supertree methods in several studies [37, 39, 25, 38, 42]. See Appendix (in the full version of the paper on bioRxiv) for full details of this study.

We explored the impact of changing the number of source trees. The result for two source trees is predicted by theory (i.e., GreedyRFS is the same as Exact-2-RFS for two source trees, and so is guaranteed optimal for this case), but even when the number of source trees was greater than two, GreedyRFS dominated FastRFS in terms of criterion score, provided that the number of source trees was not too large (Fig. 5).

This establishes that the advantage in criterion score is not limited to the case of two source trees, suggesting that using Exact-2-RFS within GreedyRFS (or some other heuristics) may be useful for supertree estimation more generally.

5 Conclusions

The main contribution of this paper is Exact-2-RFS, a polynomial time algorithm for the Robinson-Foulds Supertree (RFS) of two trees that enables divide-and-conquer pipelines to be provably statistically consistent under sequence evolution models (e.g., GTR [40] and

MSC+GTR [31]). Our experimental study showed that Exact-2-RFS dominates the leading RFS heuristic, FastRFS, when used within divide-and-conquer species tree estimation using genome-scale datasets, a problem of increasing importance in biology. We also showed that a greedy heuristic using Exact-2-RFS produced better criterion scores than FastRFS when the number of source trees was small to moderate, showing the potential for Exact-2-RFS to be useful in other settings. Overall, our study advances the theoretical understanding of several important supertree problems and also provides a new method that should improve scalability of phylogeny estimation methods.

This study suggests several directions for future work. For example, although we showed that Exact-2-RFS produced better RFS criterion scores than FastRFS when used in divide-and-conquer species tree estimation (and similarly GreedyRFS was better than FastRFS for small numbers of source trees in supertree estimation), additional studies are needed to explore its performance, including additional datasets (both simulated and biological datasets) and other leading supertree methods. Similarly, other heuristics using Exact-2-RFS besides GreedyRFS should be developed and studied. Finally, our study explored accuracy rather than computational aspects; hence, a comparison between methods with respect to running time would also help inform the choice of method, especially for large datasets, and should be studied.

References

1 Alfred V. Aho, Yehoshua Sagiv, Thomas G. Szymanski, and Jeffrey D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

2 Mukul S Bansal, J Gordon Burleigh, Oliver Eulenstein, and David Fernández-Baca. Robinson-Foulds supertrees. *Algorithms for Molecular Biology*, 5(1):18, 2010.

3 Julien Baste, Christophe Paul, Ignasi Sau, and Celine Scornavacca. Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees. *Bulletin of Mathematical biology*, 79(4):920–938, 2017.

4 Bernard R Baum. Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, pages 3–10, 1992.

5 Vincent Berry and François Nicolas. Maximum agreement and compatible supertrees. In *Annual Symposium on Combinatorial Pattern Matching*, pages 205–219. Springer, 2004.

6 Vincent Berry and François Nicolas. Improved parameterized complexity of the maximum agreement subtree and maximum compatible tree problems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(3):289–302, 2006.

7 Olaf RP Bininda-Emonds. *Phylogenetic supertrees: combining information to reveal the tree of life*. Springer Science & Business Media, 2004.

8 Ruchi Chaudhary, David Fernández-Baca, and John Gordon Burleigh. MulRF: a software package for phylogenetic analysis using multi-copy gene trees. *Bioinformatics*, 31(3):432–433, 2014.

9 James A Cotton and Mark Wilkinson. Majority-rule supertrees. *Systematic biology*, 56(3):445–452, 2007.

10 Leonardo De Oliveira Martins, Diego Mallo, and David Posada. A Bayesian supertree model for genome-wide species tree reconstruction. *Systematic biology*, 65(3):397–416, 2016.

11 Péter Erdős, Michael A Steel, Laszlo A Szekely, and Tandy J Warnow. Local quartet splits of a binary tree infer all quartet splits via one dyadic inference rule. *Computers and Artifical Intelligence*, 16(2):217–227, 1997.

12 Péter L Erdős, Michael A Steel, László A Székely, and Tandy J Warnow. A few logs suffice to build (almost) all trees (I). *Random Structures & Algorithms*, 14(2):153–184, 1999.

13 Péter L Erdös, Michael A Steel, László A Székely, and Tandy J Warnow. A few logs suffice to build (almost) all trees (II). *Theoretical Computer Science*, 221(1-2):77–118, 1999.

14 David Fernández-Baca, Sylvain Guillemot, Brad Shutters, and Sudheer Vakati. Fixed-parameter algorithms for finding agreement supertrees. *SIAM Journal on Computing*, 44(2):384–410, 2015.

15 Markus Fleischauer and Sebastian Böcker. Collecting reliable clades using the greedy strict consensus merger. *PeerJ*, 4:e2172, 2016.

16 Markus Fleischauer and Sebastian Böcker. Bad Clade Deletion supertrees: a fast and accurate supertree algorithm. *Molecular biology and evolution*, 34(9):2408–2421, 2017.

17 Sylvain Guillemot and Vincent Berry. Fixed-parameter tractability of the maximum agreement supertree problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2):342–353, 2010.

18 Anne Kupczok. Split-based computation of majority-rule supertrees. *BMC evolutionary biology*, 11(1):205, 2011.

19 Wayne P Maddison. Gene trees in species trees. *Systematic Biology*, 46(3):523–536, 1997.

20 FR McMorris and Michael A Steel. The complexity of the median procedure for binary trees. In *New Approaches in Classification and Data Analysis*, pages 136–140. Springer, 1994.

21 Siavash Mirarab, Rezwana Reaz, Md S Bayzid, Théo Zimmermann, M Shel Swenson, and Tandy Warnow. ASTRAL: genome-scale coalescent-based species tree estimation. *Bioinformatics*, 30(17):i541–i548, 2014. Special issue for ECCB (European Conference on Computational Biology), 2014.

22 Siavash Mirarab and Tandy Warnow. ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics*, 31(12):i44–i52, 2015. Special issue for ISMB 2015.

23 Erin K. Molloy and Tandy Warnow. FastMulRFS: fast and accurate species tree estimation under generic gene duplication and loss models. *Bioinformatics*, 2020. To appear, special issue for ISMB 2020; preprint available at https://www.biorxiv.org/content/10.1101/835553v3.full.

24 Serita Nelesen, Kevin Liu, Li-San Wang, C Randal Linder, and Tandy Warnow. DACTAL: divide-and-conquer trees (almost) without alignments. *Bioinformatics*, 28(12):i274–i282, 2012. Special issue for ISMB 2012.

25 Nam Nguyen, Siavash Mirarab, and Tandy Warnow. MRL and SuperFine+MRL: new supertree methods. *Algorithms for Molecular Biology*, 7(1):3, 2012.

26 Huw A Ogilvie, Joseph Heled, Dong Xie, and Alexei J Drummond. Computational performance and statistical accuracy of *BEAST and comparisons with other methods. *Systematic Biology*, 65(3):381–396, 2016.

27 Roderic DM Page. Modified mincut supertrees. In *Proceedings WABI (International Workshop on Algorithms in Bioinformatics)*, pages 537–551. Springer-Verlag, 2002.

28 Cynthia Phillips and Tandy J Warnow. The asymmetric median tree—a new model for building consensus trees. *Discrete Applied Mathematics*, 71(1-3):311–335, 1996.

29 Mark A Ragan. Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, 1(1):53–58, 1992.

30 David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1-2):131–147, 1981.

31 Sebastien Roch, Michael Nute, and Tandy Warnow. Long-branch attraction in species tree estimation: Inconsistency of partitioned likelihood and topology-based summary methods. *Systematic Biology*, 68(2):281–297, September 2018. doi:10.1093/sysbio/syy061.

32 Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

33 Charles Semple and Mike Steel. A supertree method for rooted trees. *Discrete Applied Mathematics*, 105(1-3):147–158, 2000.

34 Sagi Snir and Satish Rao. Quartets MaxCut: a divide and conquer quartets algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 7(4):704–718, 2010.

35 Alexandros Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313, 2014.

36 Mike Steel and Allen Rodrigo. Maximum likelihood supertrees. *Systematic biology*, 57(2):243–250, 2008.

37 M Shel Swenson, François Barbançon, Tandy Warnow, and C Randal Linder. A simulation study comparing supertree and combined analysis methods using SMIDGen. *Algorithms for Molecular Biology*, 5(1):8, 2010.

38 M Shel Swenson, Rahul Suri, C Randal Linder, and Tandy Warnow. An experimental study of Quartets MaxCut and other supertree methods. *Algorithms for Molecular Biology*, 6(1):7, 2011.

39 M Shel Swenson, Rahul Suri, C Randal Linder, and Tandy Warnow. SuperFine: fast and accurate supertree estimation. *Systematic Biology*, 61(2):214, 2011.

40 Simon Tavaré. Some probabilistic and statistical problems in the analysis of DNA sequences. In R.M. Miura, editor, *Lectures on mathematics in the life sciences–DNA sequences*, volume 17, pages 57–86. American Mathematical Society, Providence, RI, 1986.

41 Pranjal Vachaspati and Tandy Warnow. ASTRID: accurate species trees from internode distances. *BMC genomics*, 16(10):S3, 2015.

42 Pranjal Vachaspati and Tandy Warnow. FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization. *Bioinformatics*, 33(5):631–639, 2016.

43 Tandy Warnow. Concatenation analyses in the presence of incomplete lineage sorting. *PLOS Currents Tree of Life*, 2015. doi:10.1371/currents.tol.8d41ac0f13d1abedf4c4a59f5d17b1f7.

44 Tandy Warnow. Divide-and-conquer tree estimation: Opportunities and challenges. In *Bioinformatics and Phylogenetics: Seminal contributions of Bernard Moret*, pages 121–150. Springer, 2019.

45 Mark Wilkinson and James A Cotton. Supertree methods for building the tree of life: divide-and-conquer approaches to large phylogenetic problems. In T. R. Hodkinson and J. A. N. Parnell, editors, *Reconstructing the Tree of Life: Taxonomy and Systematics of Large and Species Rich Taxa*, pages 61–75. CRC Press, 2007. Systematics Association special volume 72.

46 Mark Wilkinson, James A. Cotton, Chris Creevey, Oliver Eulenstein, Simon R. Harris, Francois-Joseph Lapointe, Claudine Levasseur, James O. McInerney, Davide Pisani, and Joseph L. Thorley. The shape of supertrees to come: Tree shape related properties of fourteen supertree methods. *Systematic Biology*, 54(3):419–431, 2005.

47 Xilin Yu. Computing Robinson-Foulds supertree for two trees. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2019. Available online at http://hdl.handle.net/2142/105698.

48 Xilin Yu, Thien Le, Sarah Christensen, Erin K Molloy, and Tandy Warnow. Advancing divide-and-conquer phylogeny estimation. *bioRxiv*, 2020. doi:10.1101/2020.05.16.099895.

49 Chao Zhang, Maryam Rabiee, Erfan Sayyari, and Siavash Mirarab. ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19(6):153, 2018. Special issue for RECOMB-CG 2017.