AMB ALGORITHMS FOR MOLECULAR BIOLOGY

RESEARCH Open Access

Robinson-Foulds Supertrees

Mukul S Bansal[1,2], J Gordon Burleigh[3], Oliver Eulenstein[2], David Fernández-Baca[2*]

Abstract

Background: Supertree methods synthesize collections of small phylogenetic trees with incomplete taxon overlap into comprehensive trees, or supertrees, that include all taxa found in the input trees. Supertree methods based on the well established Robinson-Foulds (RF) distance have the potential to build supertrees that retain much information from the input trees. Specifically, the RF supertree problem seeks a binary supertree that minimizes the sum of the RF distances from the supertree to the input trees. Thus, an RF supertree is a supertree that is consistent with the largest number of clusters (or clades) from the input trees.

Results: We introduce efficient, local search based, hill-climbing heuristics for the intrinsically hard RF supertree problem on rooted trees. These heuristics use novel non-trivial algorithms for the SPR and TBR local search problems which improve on the time complexity of the best known (naïve) solutions by a factor of $\Theta(n)$ and $\Theta(n^2)$ respectively (where n is the number of taxa, or leaves, in the supertree). We use an implementation of our new algorithms to examine the performance of the RF supertree method and compare it to matrix representation with parsimony (MRP) and the triplet supertree method using four supertree data sets. Not only did our RF heuristic provide fast estimates of RF supertrees in all data sets, but the RF supertrees also retained more of the information from the input trees (based on the RF distance) than the other supertree methods.

Conclusions: Our heuristics for the RF supertree problem, based on our new local search algorithms, make it possible for the first time to estimate large supertrees by directly optimizing the RF distance from rooted input trees to the supertrees. This provides a new and fast method to build accurate supertrees. RF supertrees may also be useful for estimating majority-rule(-) supertrees, which are a generalization of majority-rule consensus trees.

Introduction

Supertree methods provide a formal approach for combining small phylogenetic trees with incomplete species overlap in order to build comprehensive species phylogenies, or supertrees, that contain all species found in the input trees. Supertree analyses have produced the first family-level phylogeny of flowering plants [1] and the first phylogeny of nearly all extant mammal species [2]. They have also enabled phylogenetic analyses using large-scale genomic data sets in bacteria, across eukaryotes, and within plants [3,4] and have helped elucidate the origin of eukaryotic genomes [5]. Furthermore, supertrees have been used to examine rates and patterns of species diversification [1,2], to test hypotheses regarding the structure of ecological communities [6], and to examine extinction risk in current species [7].

Although supertrees can support large-scale evolutionary and ecological analyses, there are still numerous concerns about the performance of existing supertree methods (e.g., [8-14]). In general, an effective supertree method must accurately estimate phylogenies from large data sets in a reasonable amount of time while retaining much of the phylogenetic information from the input trees.

By far the most commonly used supertree method is matrix representation with parsimony (MRP), which works by solving the parsimony problem on a binary matrix representation of the input trees [15,16]. While the parsimony problem is NP-hard, MRP can take advantage of fast and effective hill-climbing heuristics implemented in PAUP* or TNT (e.g., [17-19]). MRP heuristics often perform well in analyses of both simulated and empirical data sets (e.g., [20-22]); however, there are numerous criticisms of MRP. For example, MRP shows evidence of biases based on the shape and size of input trees [8,11], and MRP supertrees may

* Correspondence: fernande@cs.iastate.edu
[2]Department of Computer Science, Iowa State University, Ames, IA 50011, USA

contain relationships that are not supported by any of the input trees [9,12]. Furthermore, it is unclear if or why minimizing the parsimony score of a matrix representation of input trees is a good optimality criterion or should produce accurate supertrees.

Since evolutionary biologists rarely, if ever, know the true relationships for a group of species, it is difficult to assess the accuracy of supertree, or any phylogenetic, methods. One approach to evaluate the accuracy of supertrees is with simulations (e.g., [20,21]). However, simulations inherently simplify the true processes of evolution, and it is unclear how well the performance of a phylogenetic method in simulations corresponds to its performance with empirical data. Perhaps a more useful way to define the accuracy of a supertree method is to quantify the amount of phylogenetic information from the input trees that is retained in the supertree. Ideally, we want the supertree to reflect the input tree topologies as much as possible. This suggests that the supertree objective should directly evaluate the similarity of the supertree to the input trees (e.g., [11,23,24]).

Numerous metrics exist to measure the similarity of input trees to a supertree, and the Robinson-Foulds (RF) distance metric [25] is among the most widely used. In fact, numerous studies have evaluated the performance of supertree methods, including MRP, by measuring the RF distance between collections of input trees and the resulting supertrees (e.g., [11,20,21]). The RF supertree problem seeks a binary supertree that minimizes the sum of the RF distances between every rooted input tree and the supertree. The intuition behind seeking a *binary* supertree is that, in this setting, minimizing the RF distance is equivalent to maximizing the number of clusters (or clades) that are shared by the supertree and the input trees. Thus, an RF supertree is a supertree that is consistent with the largest number of clusters from the input trees. Unfortunately, as with MRP, computing RF supertrees is NP-hard [26]. In this work, we describe efficient hill-climbing heuristics to estimate RF supertrees. These heuristics allow the first large-scale estimates of RF supertrees and comparisons of the accuracy of RF supertrees to other commonly used supertree methods.

The RF distance metric between two rooted trees is defined to be a normalized count of the symmetric difference between the sets of clusters of the two trees. In the supertree setting, the input trees will often have only a strict subset of the taxa present in the supertree. Thus, a high RF distance between an input tree and a supertree does not necessarily correspond to conflicting evolutionary histories; it can also indicate incomplete phylogenetic information. Consequently, in order to compute the RF distance between an input tree which has only a strict subset of the taxa in the supertree, we first restrict the supertree to only the leaf set of the input tree. This adapted version of the RF distance is not a metric, or even a distance measure (mathematically speaking). However, for convenience, we will refer to this adapted version of the RF distance metric using the same name.

Previous work

Supertree methods are a generalization of consensus methods, in which all the input trees have the same leaf set. The problem of finding an optimal median tree under the RF distance in such a consensus setting is well-studied. In particular, it is known that the majority-rule consensus of the input trees must be a median tree [27], and it can be found in polynomial time. On the other hand, finding the optimum *binary* median tree, i.e. an RF supertree, in the consensus setting is NP-hard [26]. This implies that computing an RF supertree in general is NP-hard as well.

Our definition of RF distance between two trees where one has only a strict subset of the taxa in the other, corresponds to the distance measure used to define "majority-rule(-) supertrees" by Cotton and Wilkinson [28]. This definition restricts the larger tree to only the leaf set of the smaller tree before evaluating the RF distance. Majority-rule(-) supertrees are defined to be the strict consensus of all the optimal median trees under the RF distance. These median trees are defined similarly to RF supertrees, except that RF supertrees must be binary while the median trees can be non-binary. In general, majority-rule supertrees [28], in both their (-) and (+) variants, seek to generalize the majority-rule consensus. Indeed, majority-rule supertrees have been shown to have several desirable properties reminiscent of majority-rule consensus trees [29]. Although majority-rule supertrees and RF supertrees are both based on minimizing RF distance, they represent two different approaches to supertree construction. In particular, the RF supertree method seeks a supertree that is consistent with the largest number of clusters (clades) from the input trees, while majority-rule supertrees do not. Nevertheless, as we discuss later, RF supertrees could be used as a starting point to estimate majority-rule(-) supertrees.

The RF distance between two trees on the same size n leaf set, with leaves labeled by integers $\{1, ..., n\}$, can be computed in $O(n)$ time [30]. In fact, an $(1 + \epsilon)$-approximate value of the RF distance can be computed in sublinear time, with high probability [31].

In the case of unrooted trees, the RF distance metric is sometimes also known as the splits metric (e.g., [32]). The supertree analysis package Clann [23] provides heuristics that operate on unrooted trees and attempt to maximize the number of splits shared between the input trees and the inferred supertree. This method is called the "maximum splits-fit" method.

Local Search

We use a heuristic approach for the RF supertree problem. Local search is the basis of effective heuristics for many phylogenetic problems. These heuristics iteratively search through the space of possible supertrees guided, at each step, by solutions to some local search problem. More formally, in these heuristics, a *tree graph* (see [32,33]) is defined for the given set of input trees and some fixed tree edit operation. The node set of this tree graph represents the set of all supertrees on the given input trees. An edge is drawn between two nodes exactly if the corresponding trees can be transformed into each other by one tree edit operation. In our setting, the *cost* of a node in the graph is the RF distance between the supertree represented by that node and the given input trees. Given an initial node in the tree graph, the heuristic's task is to find a maximal-length path of steepest descent in the cost of its nodes and to return the last node on such a path. This path is found by solving the *local search problem* at every node along the path. The local search problem is to find a node with the minimum cost in the neighborhood of a given node. The neighborhood is defined by some tree edit operation, and hence, the time complexity of the local search problem depends on the tree edit operation used.

Two of the most extensively used tree edit operations for supertrees are rooted Subtree Prune and Regraft (SPR) [33-35] and rooted Tree Bisection and Reconnection (TBR) [22,33,34]. The best known (naïve) algorithms for the SPR and TBR local search problems for the RF supertree problem require $O(kn^3)$ and $O(kn^4)$ time respectively, where k is the number of input trees, and n is the number of leaves in the supertree solution.

Our Contribution

We describe efficient hill-climbing heuristics for the RF supertree problem. These heuristics are based on novel non-trivial algorithms that can solve the corresponding local search problems for both SPR and TBR in $O(kn^2)$ time, yielding speed-ups of $\Theta(n)$ and $\Theta(n^2)$ over the best known solutions respectively. These new algorithms are inspired by fast local search algorithms for the gene duplication problem [36,37]. Note that while the supertree itself must be binary, our algorithms work even if the input trees are not. We also examine the performance of the RF supertree method using four published supertree data sets, and compare its performance with MRP and the triplet supertree method [38]. We demonstrate that the new algorithms enable RF supertree analyses on large data sets and that the RF supertree method outperforms other supertree methods in finding supertrees that are most similar to the input trees based on the RF distance metric.

Basic Notation and Preliminaries

A *tree* T is a connected acyclic graph, consisting of a node set $V(T)$ and an edge set $E(T)$. T is *rooted* if it has exactly one distinguished node called the *root* which we denote by $rt(T)$. Throughout this work, the term tree refers to a rooted tree. We define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if y is a node on the path between $rt(T)$ and x. The set of minima under $\leq_T$ is denoted by $\mathcal{L}(T)$ and its elements are called *leaves*. The set of all *non-root internal nodes* of T, denoted by $I(T)$, is defined to be the set $V(T) \backslash (\mathcal{L}(T) \cup \{rt(T)\})$. If $\{x, y\} \in E(T)$ and $x \leq_T y$ then we call y the *parent* of x denoted by $pa_T(x)$ and we call x a *child* of y. The set of all children of y is denoted by $Ch_T(y)$. T is *fully binary* if every node has either zero or two children. If two nodes in T have the same parent, they are called *siblings*. The *least common ancestor* of a non-empty subset $L \subseteq V(T)$, denoted as $lca(L)$, is the unique smallest upper bound of L under $\leq_T$. The *subtree* of T rooted at node $y \in V(T)$, denoted by T_y, is the tree induced by $\{x \in V(T): x \leq y\}$. For each node $v \in I(T)$, the *cluster* $\mathcal{C}_T(v)$ is defined to be the set of all leaf nodes in T_v; i.e. $\mathcal{C}_T(v) = \mathcal{L}(T_v)$. We denote the set of all clusters of a tree T by $\mathcal{H}(T)$. Given a set $L \subseteq \mathcal{L}(T)$, let T' be the minimal rooted 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. The *symmetric difference* of two sets A and B, denoted by $A\Delta B$, is the set $(A\backslash B) \cup (B\backslash A)$. A *profile* $\mathcal{P}$ is a tuple of trees $(T_1, ..., T_k)$.

The RF Supertree Problem

Given a profile $\mathcal{P}$, we define a *supertree* on $\mathcal{P}$ to be a fully binary tree T^* where $\mathcal{L}(T^*) = \bigcup_{i=1}^{k} \mathcal{L}(T_i)$.

Definition 1 (RF Distance). *Given a profile* $\mathcal{P} = (T_1, ..., T_k)$ *and a supertree* T^* *on* $\mathcal{P}$, *we define the* RF distance *as follows*:

1. *For any* T_i, *where* $1 \leq i \leq k$, $RF(T_i, T^*) = |\mathcal{H}(T_i) \Delta \mathcal{H}(T^*[\mathcal{L}(T_i)])|$.
2. $RF(\mathcal{P}, T^*) = \sum_{i=1}^{k} RF(T_i, T^*)$
3. *Let* $\mathcal{T}$ *be the set of supertrees on* $\mathcal{P}$, *then* $RF(\mathcal{P}) = \min_{T^* \in \mathcal{T}} RF(\mathcal{P}, T^*)$.

Remark: Traditionally, the value of the RF distance, as computed above, is normalized by multiplying by 1/2. However, this does not affect the definition or computation of RF supertrees, and therefore, we do not normalize the RF distance.

Problem 1 (RF Supertree).

Instance: *A profile $\mathcal{P}$.*
Find: *A supertree T_{opt}on $\mathcal{P}$ such that RF ($\mathcal{P}$, T_{opt}) = RF ($\mathcal{P}$).*

Recall that the RF Supertree problem is NP-hard [27].

Local Search Problems

Here we first provide definitions for the re-rooting operation (denoted RR) and the TBR[22] and SPR[35] edit operations and then formulate the related local search problems that were motivated in the introduction.

Definition 2 (RR operation). *Let T be a tree and $x \in V(T)$. RR(T, x) is defined to be the tree T, if x = rt(T) or $x \in Ch(rt(T))$. Otherwise, RR(T, x) is the tree that is obtained from T by (i) suppressing rt(T), and (ii) subdividing the edge {pa(x), x} by a new root node. We define the following extension*: RR(*T*) = $\cup_{x \in V(T)}${RR(*T, x*)}.

For technical reasons, before we can define the TBR operation, we need the following definition.

Definition 3 (Planted tree). *Given a tree T, the* planted tree $\Phi(T)$ *is the tree obtained by adding a* root edge {*p, rt*(*T*)}, *where $p \notin V(T)$, to T.*

Definition 4 (TBR operation). (*See Fig.* 1) *Let T be a tree, e = (u, v) $\in$ E(T), where u = pa(v), and X, Y be the connected components that are obtained by removing edge e from T where $v \in X$ and $u \in Y$. We define* TBR_T *(v, x, y) for $x \in X$ and $y \in Y$ to be the tree that is obtained from $\Phi(T)$ by first removing edge e, then replacing the component X by* RR(*X, x*), *and then adjoining a new edge f between x' = rt*(RR(*X, x*)) *and Y as follows*:

1. *Create a new node y' that subdivides the edge (pa (y), y).*
2. *Adjoin the edge f between nodes x' and y'.*
3. *Suppress the node u, and rename x' as v and y' as u.*
4. *Contract the root edge.*

Notation. We define the following:

1. TBR_T (*v, x*) = $\cup_{y \in Y}$ {TBR_T (*v, x, y*)}
2. TBR_T (*v*) = $\cup_{x \in X}$ TBR_T (*v, x*)
3. TBR_T = $\cup_{(u, v) \in E(T)}$ TBR_T(*v*)

Definition 5 (SPR operation). *Let T be a tree, e = (u, v) $\in$ E(T), where u = pa(v), and X, Y be the connected components that are obtained by removing edge e from T where $v \in X$ and $u \in Y$. We define* SPR_T *(v, y), for $y \in Y$, to be the tree* TBR_T *(v, v, y). We say that the tree* SPR_T *(v, y) is obtained from T by a* subtree prune and regraft (SPR) *operation that* prunes *subtree T_v and* regrafts *it above node y.*

Notation. We define the following:

1. SPR_T (*v*) = $\cup_{y \in Y}${SPR_T (*v, y*)}
2. SPR_T = $\cup_{(u, v) \in E(T)}$ SPR_T(*v*)

Note that an SPR operation for a given tree *T* can be briefly described through the following four steps: (i) prune some subtree *P* from *T*, (ii) add a root edge to the remaining tree *S*, (iii) regraft *P* into an edge of the remaining tree *S*, and (iv) contract the root edge.

We now define the relevant local search problems based on the TBR and SPR operations.

Problem 2 (TBR-Scoring (TBR-S)). *Given instance ⟨ $\mathcal{P}$, T⟩, where $\mathcal{P}$ is the profile (T_1, ..., T_k) and T is a supertree on $\mathcal{P}$, find a tree $T^* \in$* TBR_T *such that* $RF(\mathcal{P}, T^*) = \min_{T' \in \text{TBR}_T} RF(\mathcal{P}, T')$.

Problem 3 (TBR-Restricted Scoring (TBR-RS)). *Given instance ⟨ $\mathcal{P}$, T, v⟩, where $\mathcal{P}$ is the profile (T_1, ..., T_k), T is a supertree on $\mathcal{P}$, and v is a non-root node in V(T), find a tree T * $\in$* TBR_T *(v) such that* $RF(\mathcal{P}, T^*) = \min_{T' \in \text{TBR}_T(v)} RF(\mathcal{P}, T')$.

The problems SPR-*Scoring* (SPR-*S*) and SPR-*Restricted Scoring* (SPR-*RS*) are defined analogously to the problems TBR-S and TBR-RS respectively.

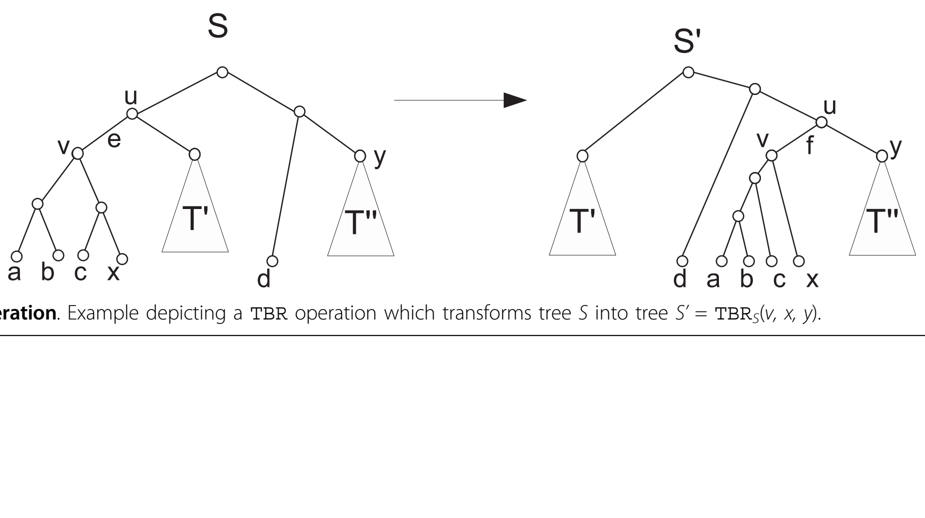

Figure 1 TBR Operation. Example depicting a TBR operation which transforms tree *S* into tree *S'* = TBR_S(*v, x, y*).

Throughout the remainder of this manuscript, k is the number of trees in the profile $\mathcal{P}$, T denotes a supertree on $\mathcal{P}$, and n is the number of leaves in T. The following observation follows from Definition 4.

Observation 1. *The* TBR-S *problem on instance* $\langle \mathcal{P}, T \rangle$ *can be solved by solving the* TBR-RS *problem* $|E(T)|$ *times.*

We show how to solve the TBR-S problem on the instance $\langle \mathcal{P}, T \rangle$ in $O(kn^2)$ time. Since $\text{SPR}_T \subseteq \text{TBR}_T$ this also implies an $O(kn^2)$ solution for the SPR-S problem. This gives a speed-up of $\Theta(n^2)$ and $\Theta(n)$ over the best known (naïve) algorithms for the TBR-S and SPR-S problems respectively.

In particular, we first show that any instance of the TBR-RS problem can be decomposed into an instance of an SPR-RS problem, and an instance of a Rooting problem (defined in the next section). We show how to solve both these problems in $O(kn)$ time, yielding an $O(kn)$ time solution for the TBR-RS problem. This immediately implies an $O(kn^2)$ time algorithm for the TBR-S problem (see Observation 1).

Note that the size of the set TBR_T is $\Theta(n^3)$. Thus, for each tree in the input profile the time complexity of computing and enumerating the RF distances of all trees in TBR_T is $\Omega(n^3)$. However, to solve the TBR-S problem one only needs to find a tree with the minimum RF distance. This lets us solve the TBR-S problem in time that is sub-linear in the size of TBR_T. In fact, after the initial $O(kn^2)$ preprocessing step, our algorithm can output the RF distance of any tree in TBR_T in $O(1)$ time.

Structural Properties

Throughout this section, we limit our attention to one tree S from the profile $\mathcal{P}$. We show how to solve the TBR-RS problem for the instance $\langle (S), T, v \rangle$ for some non-root node $v \in V(T)$ in $O(n)$ time. Based on this solution, it is straightforward to solve the TBR-RS problem on the instance $\langle \mathcal{P}, T, v \rangle$ with-in $O(kn)$ time as well. For clarity, we will also assume that $\mathcal{L}(S) = \mathcal{L}(T)$. In general, if $\mathcal{L}(S) \subset \mathcal{L}(T)$ then we can simply set T to be $T[\mathcal{L}(S)]$. This takes $O(n)$ time and, consequently, does not affect the time complexity of our algorithm.

Our algorithm makes use of the LCA mapping from S to T. This mapping is defined as follows.

Definition 6 (LCA Mapping). *Given two trees* T' and T *such that* $\mathcal{L}(T') \subseteq \mathcal{L}(T)$, *the* LCA mapping $\mathcal{M}_{T', T}: V(T') \to V(T)$ *is the mapping* $\mathcal{M}_{T', T}(u) = lca_T(\mathcal{L}(T'_u))$.

Notation. We define a boolean function $f_T: I(S) \to \{0, 1\}$ such that $f_T(u) = 1$ if there exists a node $v \in I(T)$ such that $\mathcal{C}_S(u) = \mathcal{C}_T(v)$, and $f_T(u) = 0$ otherwise. Thus, $f_T(u) = 1$ if and only if the cluster $\mathcal{C}_S(u)$ exists in the tree T as well. Additionally, we define $\mathscr{F}_T = \{u \in I(S): f_T(u) = 0\}$; that is, $\mathscr{F}_T$ is the set of all nodes $u \in I(S)$ such that the cluster $\mathcal{C}_S(u)$ does not exist in the tree T.

The following lemma associates the value $RF(S, T)$ with the cardinality of the set $\mathscr{F}_T$.

Lemma 1. $RF(S, T) = |I(T)| - |I(S)| + 2 \cdot |\mathscr{F}_T|$.

Proof. Let $\mathcal{G}_T$ denote the set $\{u \in I(S): f_T(u) = 1\}$. By the definition of $RF(S, T)$, we must have $RF(S, T) = |I(T)| + |I(S)| - 2 \cdot |\mathcal{G}_T|$. And hence, since $|\mathcal{G}_T| + |\mathscr{F}_T| = I(S)$, we get $RF(S, T) = |I(T)| - |I(S)| + 2 \cdot |\mathscr{F}_T|$. □

Lemma 2. *For any* $u \in I(S)$, $f_T(u) = 1$ *if and only if* $|\mathcal{C}_S(u)| = |\mathcal{C}_T(\mathcal{M}_{S,T}(u))|$.

Proof. If $|\mathcal{C}_S(u)| = |\mathcal{C}_T(\mathcal{M}_{S,T}(u))|$ then we must have $\mathcal{C}_S(u) = \mathcal{C}_T(\mathcal{M}_{S,T}(u))$ and, consequently, $f_T(u) = 1$. In the other direction, if $|\mathcal{C}_S(u)| \neq |\mathcal{C}_T(\mathcal{M}_{S,T}(u))|$, then we must have $\mathcal{C}_S(u) \subset \mathcal{C}_T(\mathcal{M}_{S,T}(u))$ and, consequently, $f_T(u) = 0$. □

The LCA mapping from S to T can be computed in $O(n)$ time [39], and consequently, by Lemmas 1 and 2, we can compute the RF distance between S and T in $O(n)$ time as well (other $O(n)$-time algorithms for calculating the RF distance are presented in [30,31]). Moreover, Lemma 1 implies that in order to find a tree $T^* \in \text{TBR}_T(v)$ such that $RF(\mathcal{P}, T^*) = \min_{T' \in \text{TBR}_T(v)} RF(\mathcal{P}, T')$, it is sufficient to find a tree $T^* \in \text{TBR}_T(v)$ for which $|\mathcal{F}_{T^*}| = \min_{T' \in \text{TBR}_T(v)} |\mathcal{F}_{T'}|$.

Remark: An implicit assumption here is that the leaves of both trees are labeled by integers $\{1, ..., n\}$. If the leaf labels are arbitrary, then we require an additional $O(kn \log n)$-time preprocessing step to relabel the leaves of the trees in the given profile. Note, however, that this additional step does not add to the overall time complexity of solving the TBR-S or SPR-S problems.

We now show that the TBR-RS problem can be solved by solving two smaller problems separately and combining their solutions.

As before, we limit our attention to one tree S from the profile $\mathcal{P}$. Given the TBR-RS instance $\langle (S), T, v \rangle$, we define a bipartition $\{X, \overline{X}\}$ of $I(S)$, where $X = \{u \in I(S): \mathcal{M}_{S,T}(u) \in V(T_v)\}$.

Lemma 3. *If* $u \in X$, *then* $f_{T'}(u) = f_T(u)$ *for all* $T' \in \text{TBR}_T(v, v)$. *If* $u \in \overline{X}$ *and* y *denotes the sibling of* v, *then* $f_{T'}(u) = f_T(u)$, *where* $T' = \text{TBR}_T(v, x, y)$ *for any* $x \in V(T_v)$.

Proof. Consider the case when $u \in X$. Let T' be any tree in $\text{TBR}_T(v, v)$ and let node $y \in V(T)$ be such that $T' = \text{TBR}(v, v, y)$. Thus, for any node $w \in V(T_v)$, the subtrees T_v and T'_v must be identical. Since $u \in X$, we must have $\mathcal{M}_{S,T}(u) \in T_v$ and, consequently, $T'_{\mathcal{M}_{S,T'}(u)} = T_{\mathcal{M}_{S,T}(u)}$. Lemma 2 now implies that $f_{T'}(u) = f_T(u)$.

Now consider the case when $u \in \overline{X}$. Node y denotes the sibling of v in tree T and let $T' = \text{TBR}(v, x, y)$, for some $x \in V(T_v)$. Thus, for any node $w \in V(T) \backslash V(T_v)$, we must have $\mathcal{L}_T(w) = \mathcal{L}_{T'}(w)$. Moreover, the leaf sets of the two subtrees rooted at the children of w in T must be identical to the leaf sets of the two subtrees rooted at the children of w in T': This implies that if $\mathcal{M}_{S,T}(u) = w$, then $\mathcal{M}_{S,T'}(u) = w$ as well. By Lemma 2 we must therefore have $f_{T'}(u) = f_T(u)$. □

Lemma 3 implies that a tree in $\mathtt{TBR}_T(v)$ with smallest RF distance can be obtained by optimizing the rooting for the pruned subtree, and optimizing the regraft location separately. This allows us to obtain a tree in $\mathtt{TBR}_T(v)$ with smallest RF distance by evaluating only $O(n)$ trees. Contrast this with the naïve approach to finding a tree in $\mathtt{TBR}_T(v)$ with smallest total distance, which is to evaluate all trees obtained by rerooting the pruned subtree in all possible ways, and, for each rerooting, regrafting the subtree in all possible locations. Since there are $O(n)$ ways to reroot the pruned subtree, and $O(n)$ ways to regraft, this would require evaluating $O(n^2)$ trees. It is interesting to note that this ability to decompose the TBR-RS problem into two simpler problems is not unique to the context of RF supertrees alone. For example, it has been observed that a similar decomposition can be achieved in the context of the gene duplication problem [37].

Thus, to solve the TBR-RS problem, we must find (i) a rerooting T' of the subtree T_v for which $\mathscr{F}_{T'}$ is minimized, and (ii) a regraft location y for T_v which minimizes $|\mathscr{F}_{\mathtt{SPR}(v,\,y)}|$. Observe that the problem in part (ii) is simply the SPR-RS problem on the input instance $\langle (S), T, v\rangle$. For part (i), consider the following problem statement.

Problem 4 (Rooting). *Given instance $\langle \mathcal{P}, T, v\rangle$, where $\mathcal{P}$ is the profile $(T_1, ..., T_k)$, T is a supertree on $\mathcal{P}$, and v is a non-root node in $V(T)$, find a node $x \in V(T_v)$ for which RF $(\mathcal{P}, \mathtt{TBR}_T(v, x, y))$ is minimum, where y denotes the sibling of v in T.*

Note that the problem in part (i) is the Rooting problem on the input instance $\langle (S), T, v\rangle$. We show how to solve both the Rooting and the SPR-RS problems in $O(n)$ time on instance $\langle (S), T, v\rangle$. As seen above, based on Lemma 3, this immediately implies that the TBR-RS problem for a profile consisting of a single tree can be solved in $O(n)$ time. To solve the TBR-RS problem on instance $\langle \mathcal{P}, T, v\rangle$, we simply solve the Rooting and SPR-RS problems separately on the input instance $\langle \mathcal{P}, T, v\rangle$, which takes $O(kn)$ time (see Theorems 3 and 4). We thus have the following two theorems.

Theorem 1. *The* TBR*-RS problem can be solved in $O(kn)$ time.*

Theorem 2. *The* TBR*-S problem can be solved in $O(kn^2)$ time.*

Solving the Rooting Problem

To solve the Rooting problem on instance $\langle (S), T, v\rangle$, we rely on an efficient algorithm for computing the value of $f_{T'}(u)$ for any $T' \in \mathtt{RR}(T_v)$ and any $u \in I(S)$. This algorithm relies on the following five lemmas. Let a denote the node $\mathcal{M}_{S,T}(u)$, y denote the sibling of v in T, and $T' = \mathtt{TBR}_T(v, x, y)$ for $x \in V(T_v)$. Depending on a and $f_T(u)$ there are five possible cases: (i) $a \notin V(T_v)$, (ii) $a = rt(T_v)$ and $f_T(u) = 1$, (iii) $a = rt(T_v)$ and $f_T(u) = 0$, (iv) $a \in V(T_v) \backslash rt(T_v)$ and $f_T(u) = 1$, and (v) $a \in V(T_v) \backslash rt(T_v)$ and $f_T(u) = 0$. Lemmas 4 through 8 characterize the value $f_{T'}(u)$ for each of these five cases respectively.

Lemma 4. *If $a \notin V(T_v)$, then $f_{T'}(u) = f_T(u)$ for any $x \in V(T_v)$.*

Proof. Follows directly from Lemma 3. □

Lemma 5. *If $a = rt(T_v)$ and $f_T(u) = 1$, then $f_{T'}(u) = 1$ for all $x \in V(T_v)$.*

Proof. Since we have $a = rt(T_v)$ and $f_T(u) = 1$, by Lemma 2 we must have $\mathcal{L}(S_u) = \mathcal{L}(T_v)$. Thus, for any $x \in V(T_v)$, $\mathcal{M}_{S,T'}(u)$ must be the root of the subtree RR (T_v, x). The lemma follows. □

Lemma 6. *Let L denote the set $\mathcal{L}(T_v) \backslash \mathcal{L}(S_u)$, and let $b = lca_{T_v}(L)$. If $a = rt(T_v)$ and $f_T(u) = 0$, then,*

1. *for $x \not\leq_{T_v} b$, $f_{T'}(u) = 0$, and,*
2. *for $x \leq_{T_v} b$, $f_{T'}(u) = 1$ if and only if $|L| = |\mathcal{L}(T_b)|$.*

Proof. Since $a = rt(T_v)$ and $f_T(u) = 0$, by Lemma 2 we must have $\mathcal{L}(S_u) \neq \mathcal{L}(T_v)$. We analyze each part of the lemma separately.

1. $x \not\leq_{T_v} b$: For this case to be valid, we must have $b <_{T_v} rt(T_v)$. Therefore, let $b' = pa_{T_v}(b)$. For any T' in this case, $b' = pa_{T'}(b)$. Moreover, $\mathcal{L}(T'_{b'}) \cap \mathcal{L}(S_u) \neq \emptyset$. Therefore, we must have $b <_{T'} \mathcal{M}_{S,T'}(u)$ Hence, $\mathcal{L}(S_u) \neq \mathcal{L}(T_{\mathcal{M}_{S,T'}(u)})$ in this case, and, consequently, Lemma 2 implies that $f_{T'}(u) = 0$.
2. $x \leq_{T_v} b$: We divide our analysis into two cases:
   (a) $|L| = |\mathcal{L}(T_b)|$: In this case we must have $b \neq rt(T_v)$. Therefore, let b' denote the parent of b in tree T_v. Now consider the tree T'. The set $\mathcal{L}(T'_{b'})$ must be identical to $\mathcal{L}(S_u)$. Hence, $f_{T'}(u) = 1$ in this case.
   (b) $|L| \neq |\mathcal{L}(T_b)|$: We claim that there does not exist any edge $(pa(w), w) \in E(T_v)$ such that $\mathcal{L}(T_w)$ is either $\mathcal{L}(S_u)$ or L. Let us suppose, for the sake of contradiction, that such an edge exists. If $\mathcal{L}(T_w) = \mathcal{L}(S_u)$ then we must have $a = w$, which is a contradiction since $a = rt(T_v)$. If $\mathcal{L}(T_w) =$ g L then we must have $b = w$, and, consequently, $|L| \neq |\mathcal{L}(T_b)|$, which is, again, a contradiction. Thus, such an edge $(pa(w), w) \in E(T_v)$ cannot exist. Hence, we must have $f_{T'}(u) = 0$ for every $x \in V(T_v)$ in this case.

The lemma follows. □

Lemma 7. *If $a \in V(T_v) \backslash rt(T_v)$ and $f_T(u) = 1$, then $f_{T'}(u) = 0$ if and only if $x < T_v\ a$.*

Proof. By Lemma 2 we must have $\mathcal{L}(S_u) = \mathcal{L}(T_a)$. We have two cases:

1. $x \not<_{T_v} a$: In this case we must have $\mathcal{M}_{S,\ T'}(u) = a$, and $\mathcal{L}(T_a) = \mathcal{L}(T'_a)$. Thus, $\mathcal{L}(S_u) = \mathcal{L}(T'_a)$ and hence, $f_{T'}(u) = 1$.
2. $x <_{T_v} a$: In this case, $\mathcal{M}_{S,\ T'}(u)$ must be the root of the subtree $\mathrm{RR}(T_v, x)$. Since $\mathcal{L}(\mathrm{RR}(T_v, x)) = \mathcal{L}(T_v)$, and $\mathcal{L}(S_u) \neq \mathcal{L}(T_v)$, Lemma 2 implies that $f_{T'}(u) = 0$.

The lemma follows. □

Lemma 8. *If* $a \in V(T_v)\backslash rt(T_v)$ *and* $f_T(u) = 0$, *then* $f_{T'}(u) = 0$ *for all* $x \in V(T_v)$.

Proof. By Lemma 2 we must have $\mathcal{L}(S_u) \neq \mathcal{L}(T_a)$. We have two possible cases:

1. $x \not<_{T_v} a$: In this case we must have $\mathcal{M}_{S,\ T'}(u) = a$, and $\mathcal{L}(T_a) = \mathcal{L}(T'_a)$. Thus, $\mathcal{L}(S_u) \neq \mathcal{L}(T'_a)$ and hence, $f_{T'}(u) = 0$
2. $x <_{T_v} a$: In this case, $\mathcal{M}_{S,\ T'}(u)$ must be the root of the subtree $\mathrm{RR}(T_v, x)$. Since $\mathcal{L}(\mathrm{RR}(T_v, x)) = \mathcal{L}(T_v)$, and $\mathcal{L}(S_u) \neq \mathcal{L}(T_v)$, Lemma 2 implies that $f_{T'}(u) = 0$.

The lemma follows.

The Algorithm. For any $x \in V(T_v)$ let $A(x)$ denote the cardinality of the set
$\{u \in I(S): f_T(u) = 0, \text{ but } f_{T'}(u) = 1\}$, and $B(x)$ the cardinality of the set
$\{u \in I(S): f_T(u) = 1, \text{ but } f_{T'}(u) = 0\}$, where $T' = \mathrm{TBR}_T(v, x, y)$.

By definition, to solve the Rooting problem we must find a node $x \in V(T_v)$ for which $|A(x)| - |B(x)|$ is maximized. Our algorithm computes, at each node $x \in V(T_v)$, the values $A(x)$ and $B(x)$.

In a preprocessing step, our algorithm computes the mapping $\mathcal{M}_{S,\ T}$ as well as the size of each cluster in S and T, and creates and initializes (to 0) two counters $\alpha(x)$ and $\beta(x)$ at each node $x \in V(T_v)$. This takes $O(n)$ time. When the algorithm terminates, the values $\alpha(x)$ and $\beta(x)$ at any $x \in V(T_v)$ will be the values $\alpha(x)$ and $\beta(x)$.

Recall that, given $u \in I(S)$, a denotes the node $\mathcal{M}_{S,\ T}(u)$. Thus, any given $u \in I(S)$ must satisfy the precondition (given in terms of a) of exactly one of the the Lemmas 4 through 8. Moreover, the precondition of each of these lemmas can be checked in $O(1)$ time.

The algorithm then traverses through S and considers each node $u \in I(S)$. There are three cases:

1. If u satisfies the preconditions of Lemmas 4, 5, or 8 then we must have $f_{T'}(u) = f_T(u)$. Consequently, we do nothing in this case.
2. If u satisfies the precondition of Lemma 7, then we increment the value of $\beta(x)$ at each node $x \in V(T_a)\backslash\{a\}$ (where a is as in the statement of Lemma 7). To do this efficiently we can simply increment a counter at node a such that, after all $u \in I(S)$ have been considered, a single pre-order traversal of T_v can be used to compute the correct values of $\beta(x)$ at each $x \in V(T_v)$.
3. If u satisfies the precondition of Lemma 6, then we proceed as follows: Let a and L be as in the statement of Lemma 6. According to the Lemma, if we can find a node $b \in V(T_v)$ such that $b = lca_{T_v}(\mathrm{L})$ and $|\mathcal{L}(T_b)| = |L|$, then we increment the value of $\alpha(x)$ at each node $x \in V(T_b)$; otherwise, if such a b does not exist, we do nothing. As before, to do this efficiently, we only increment a single counter at node b such that, after all $u \in I(S)$ have been considered, a pre-order traversal of T_v suffices to compute the correct values of $\alpha(x)$ at each $x \in V(T_v)$. In order to prove the $O(n)$ run-time for this algorithm we will now explain how to precompute such a corresponding node b (if it exists), for each $u \in I(S)$ satisfying the precondition of Lemma 6, within $O(n)$ time. Note that any edge in a tree bi-partitions its leaf set. Construct the tree $S' = S[\mathcal{L}(T_v)]$. Observe that, given any candidate u, the corresponding node b exists if and only if the partition of $\mathcal{L}(S')$ induced by the edge $(u, pa(u))$ $E(S')$, is also induced by some edge, e, in the tree T_v If such an e exists, then b must be that node on e which is farther away from the root, i.e. the edge e must be the edge $(b, pa(b))$ in T_v This edge e (or its absence) can be precomputed, for all candidate u, as follows: Compute the strict consensus of the unrooted variants of the trees S' and T_v. Every edge in this strict consensus corresponds to an edge in S' and an edge in T_v that induce the same bi-partitions in the two trees.

Thus, for all candidate u that lie on such an edge, the corresponding node b can be inferred in $O(1)$ time (by using the association between the edges of the strict consensus and the edges of S' and T_v), and for all candidate u that do not lie on such an edge, we know that the corresponding node b does not exist. This strict consensus of the unrooted variants of S' and T_v can be precomputed with-in $O(n)$ time by using the algorithm of Day [30].

Hence, the Rooting problem for a profile consisting of a single tree can be solved in $O(n)$ time; yielding the following theorem.

Theorem 3. *The* Rooting *problem can be solved in* $O(kn)$ *time.*

Solving the SPR-RS Problem

We will show how to solve the SPR-RS problem on instance $\langle(S), T, v\rangle$ in $O(n)$ time. Consider the tree $R = \mathrm{SPR}_T(v, rt(T))$ Observe that, since $\mathrm{SPR}_R(v) = \mathrm{SPR}_S(v)$, solving the SPR-RS problem on instance $\langle(S), T, v\rangle$ is equivalent to solving it on the instance $\langle(S), R, v\rangle$. Thus, in the remainder of this section, we will work with tree R instead of tree S. The following four lemmas let us efficiently infer, for any $u \in I(S)$, whether $f_{T'}(u) = 1$ or $f_{T'}(u) = 0$, for any given $T' \in \mathrm{SPR}_R(v)$.

For brevity, let a denote the node $\mathcal{M}_{S,R}(u)$, and let Q denote the set $V(R)\backslash(V(R_v) \cup \{rt(R)\})$. Let $T' = \mathrm{SPR}_R(v, x)$, for any $x \in Q$.

Depending on a and $f_R(u)$ there are four possible cases: (i) $a \in V(R_v)$, (ii) $a \in Q$ and $f_R(u) = 1$, (iii) $a \in Q$ and $f_R(u) = 0$, and (iv) $a = rt(R)$. Lemmas 9 through 12 characterize the value $f_{T'}(u)$ for each of these four cases respectively. □

Lemma 9. *If* $a \in V(R_v)$, *then* $f_{T'}(u) = f_R(u)$ *for any* $x \in Q$.

Proof. Observe that $\mathrm{TBR}_R(v, v) = \mathrm{SPR}_R(v)$. Lemma 3 now immediately completes the proof. □

Lemma 10. *If* $a \in Q$ *and* $f_R(u) = 1$, *then,*

1. $f_{T'}(u) = 0$, *for* $x <_R a$, *and*
2. $f_{T'}(u) = 1$, *otherwise.*

Proof. Since $f_R(u) = 1$, Lemma 2 implies that $|\mathcal{C}_S(u)| = |\mathcal{C}_R(a)|$. Let $T' = \mathrm{SPR}_R(v, x)$; we now have two cases.

1. $x <_R a$: In this case $\mathcal{M}_{S,T'}(u) = a$, and, since $|\mathcal{C}_S(u)| < |\mathcal{C}_R(a)| < |\mathcal{C}_{T'}(a)|$, we must have $f_{T'}(u) = 0$ (by Lemma 2).
2. $x \nless_R a$: In this case $\mathcal{M}_{S,T'}(u) = a$, and since $|\mathcal{C}_S(u)| = |\mathcal{C}_R(a)| = |\mathcal{C}_{T'}(a)|$, we must have $f_{T'}(u) = 1$ (by Lemma 2).

The lemma follows. □

Lemma 11. *If* $a \in Q$ *and* $f_R(u) = 0$, *then* $f_{T'}(u) = 0$ *for any* $x \in Q$.

Proof. Since $f_R(u) = 0$, Lemma 2 implies that $|\mathcal{C}_S(u)| \neq |\mathcal{C}_R(a)|$. Thus, by the definition of LCA mapping, $|\mathcal{C}_S(u)| < |\mathcal{C}_R(a)|$. Let $T' = \mathrm{SPR}_R(v, x)$; we now have two cases.

1. $x <_R a$: In this case $\mathcal{M}_{S,T'}(u) = a$, and, since $|\mathcal{C}_S(u)| < |\mathcal{C}_R(a)| < |\mathcal{C}_{T'}(a)|$, we must have $f_{T'}(u) = 0$ (by Lemma 2).
2. $x \nless_R a$: In this case $\mathcal{M}_{S,T'}(u) = a$, and since $|\mathcal{C}_S(u)| < |\mathcal{C}_R(a)| = |\mathcal{C}_{T'}(a)|$, we must have $f_{T'}(u) = 0$ (by Lemma 2).

The lemma follows. □

For the next lemma, let S' be the tree obtained from S by suppressing all nodes s for which $\mathcal{M}_{S,R}(s) \in R_v$.

Lemma 12. *If* $a = rt(R)$ *and* $b = \mathcal{M}_{S',R}(u)$, *then,* $f_{T'}(u) = 1$ *if and only if* $x <_R b$ *and* $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| = |\mathcal{L}(S_u)|$.

Proof. First, observe that, since $a = rt(R)$, the mapping $\mathcal{M}_{S',R}(u)$ is well defined. Second, since $b = \mathcal{M}_{S',R}(u)$, we must have $\mathcal{L}(S'_u) \subseteq \mathcal{L}(R_b)$, which implies that $\mathcal{L}(S_u) \subseteq \mathcal{L}(R_v) \subseteq \mathcal{L}(R_b)$. We now have the following three cases:

1. $x \nless_R b$: In this case we must have $\mathcal{M}_{S,T'}(u) = lca_{T'}(x, b)$. By Lemma 2 we know that $f_{T'}(u) = 1$ only if $|\mathcal{C}_S(u)| = |\mathcal{C}_{T'}(\mathcal{M}_{S,T'}(u))|$. However, since we have $\mathcal{L}(S_u) \subseteq \mathcal{L}(R_v) \subseteq \mathcal{L}(R_b)$, and $x \nless_R b$, we must have $|\mathcal{C}_S(u)| < |\mathcal{C}_{T'}(\mathcal{M}_{S,T'}(u))|$; and hence, $f_{T'}(u) = 0$.
2. $x <_R b$ **and** $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| \neq |\mathcal{L}(S_u)|$: In this case we must have $\mathcal{M}_{S,T'}(u) = b$. Since $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| \neq |\mathcal{L}(S_u)|$, we must have $\mathcal{L}(S_u) \subset \mathcal{L}(R_v) \cup \mathcal{L}(R_b)$, which implies that $|\mathcal{C}_S(u)| < |\mathcal{C}_{T'}(\mathcal{M}_{S,T'}(u))|$. Thus, by Lemma 2, we must have $f_{T'}(u) = 0$.
3. $x <_R b$ **and** $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| = |\mathcal{L}(S_u)|$: In this case we must have $\mathcal{M}_{S,T'}(u) = b$. Moreover, since $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| = |\mathcal{L}(S_u)|$, we must have $|\mathcal{C}_S(u)| = |\mathcal{C}_{T'}(\mathcal{M}_{S,T'}(u))|$. Thus, by Lemma 2, we must have $f_{T'}(u) = 1$.

The lemma follows. □

The Algorithm. Note that $\mathrm{SPR}_T(v) = \mathrm{SPR}_R(v) = \cup_{x \in Q} \mathrm{SPR}_R(v, x)$. For any $x \in Q$, let $A(x) = |\{u \in I(S): f_R(u) = 0$, but $f_{T'}(u) = 1\}|$, and $B(x) = |\{u \in I(S): f_R(u) = 1$, but $f_{T'}(u) = 0\}|$, where $T' = \mathrm{SPR}_R(v, x)$. By definition, to solve the SPR-RS problem on instance $\langle(S), T, v\rangle$ we must find a node $x \in Q$ for which $|A(x)| - |B(x)|$ is maximized. Our algorithm computes, at each node $x \in Q$, the values $A(x)$ and $B(x)$.

In a preprocessing step, our algorithm first constructs the tree R computes the mapping $\mathcal{M}_{S,R}$ as well as the size of each cluster in S and R, and creates and initializes (to 0) two counters $\alpha(x)$ and $\beta(x)$ at each node $x \in Q$. This takes a total of $O(n)$ time. When the algorithm terminates, the values $\alpha(x)$ and $\beta(x)$, at any $x \in Q$ will be the values $A(x)$ and $B(x)$.

Recall that, given $u \in I(S)$, a denotes the node $\mathcal{M}_{S,R}(u)$. Thus, any given $u \in I(S)$ must satisfy the precondition (given in terms of a) of exactly one of the the Lemmas 9 through 12. Moreover, the precondition of each of these lemmas can be checked in $O(1)$ time.

The algorithm then traverses through S and considers each node $u \in I(S)$. There are three cases:

1. If u satisfies the preconditions of Lemmas 9 or 11 then we must have $f_{T'}(u) = f_R(u)$ Consequently, we do nothing in this case.
2. If u satisfies the precondition of Lemma 10, then we increment the value of $\beta(x)$ at each node $x \in V(T_a)\backslash\{a\}$ (where a is as in the statement of Lemma

10). This can be done efficiently as shown in part (2) of the algorithm for the Rooting problem.

3. If u satisfies the precondition of Lemma 6, and if $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| = |\mathcal{L}(S_u)|$, then we increment the value of $\alpha(x)$ at each node $x \in V(T_b)\setminus\{b\}$ (where a and b are as in the statement of Lemma 6).

Again, to do this efficiently, we increment a counter at node b, and perform a subsequent pre-order traversal. Note also that the mapping $\mathcal{M}_{S', R}$ can be computed in $O(n)$ time in the preprocessing step, and hence the node b can be inferred in $O(1)$ time. The condition $|\mathcal{L}(R_b)| + |\mathcal{L}(R_v)| = |\mathcal{L}(S_u)|$ is also verifiable in $O(1)$ time.

Hence, the SPR-RS problem for a profile consisting of a single tree can be solved in $O(n)$ time; yielding the following theorem.

Theorem 4. *The* SPR*-RS problem can be solved in* $O(kn)$ *time.*

Remark. To improve the performance of local search heuristics in phylogeny construction, the starting tree for the first local search step is often constructed using a greedy 'stepwise addition' procedure. This greedy procedure builds a starting species tree step-by-step by adding one taxon at a time at its locally optimal position. In the context of RF supertrees, our algorithm for the SPR-RS problem also yields a $\Theta(n)$ speed-up over naïve algorithms for this greedy procedure.

Experimental Evaluation

In order to evaluate the performance of the RF supertree method, we implemented an RF heuristic based on the SPR local search algorithm. We focused on the SPR local search because it is faster and simpler to implement than TBR, and in analyses of MRF and triplet supertrees, the performance of SPR and TBR was very similar [22,38]. We compared the performance of the RF supertree heuristic to MRP and the triplet supertree method (which seeks a supertree with the most shared triplets with the collection of input trees) using published supertree data sets from sea birds [40], marsupials [41], placental mammals [42], and legumes [43]. The published data sets contain between 7 and 726 input trees and between 112 and 571 total taxa (Table 1).

There are a number of ways to implement any local search algorithm. Preliminary analyses of the RF heuristic based on the SPR local search indicated that, as with other phylogenetic methods, the starting tree can affect the estimate of the final supertree. Occasionally the SPR searches got caught in local optima with relatively high RF-distance scores. To ameliorate this potential problem, we implemented a ratchet search heuristic for RF supertrees based on the parsimony ratchet [44]. In general, a ratchet search performs a number of iterations – in our case 25 – that consist of two local SPR searches: one in which the characters (input trees) are equally weighted, and another in which the set of the characters are re-weighted. We re-weighted the characters by randomly removing approximately two-thirds of the input trees. The goal of re-weighting the characters is to alter the tree space to avoid getting caught in a globally suboptimal part of the tree space. At the end of each iteration, the best tree is taken as the starting point of the next iteration. For each data set, we started RF ratchet searches from 20 random sequence addition starting trees, and we also ran three replicates starting from an optimal MRP supertree. All RF supertree analyses were performed on an 3 GHz Intel Pentium 4 based desktop computer with 1 GB of main memory. The RF-ratchet runs took between 5 seconds (for the Sea Birds data set) and 90 minutes (for the legume dataset) when starting from a random sequence addition tree. RF-ratchet runs starting from optimal MRP trees were at least twice as fast because they required fewer search steps.

For our MRP analyses, we also tried two heuristic search methods, both implemented using PAUP* [18]. First, we performed 20 replicates of TBR branch swapping from trees built with random addition sequence

Table 1 Experimental Results

Data Set	Supertree Method	RF-Distance	Parsimony Score
Marsupial (272 taxa; 158 trees)	RF-Ratchet	1514	2528
	RF-MRP	**1502**	2513
	MRP-TBR	1514	**2509**
	MRP-Ratchet	1514	**2509**
	Triplet	1604	2569
Sea Birds (121 taxa; 7 trees)	RF-Ratchet	**61**	223
	RF-MRP	**61**	223
	MRP-TBR	63	**221**
	MRP-Ratchet	63	**221**
	Triplet	**61**	223
Placental Mammals (116 taxa; 726 trees)	RF-Ratchet	**5686**	8926
	RF-MRP	5690	8890
	MRP-TBR	5694	**8878**
	MRP-Ratchet	5694	**8878**
	Triplet	6032	9064
Legumes (571 taxa; 22 trees)	RF-Ratchet	1556	965
	RF-MRP	**1534**	882
	MRP-TBR	1554	856
	MRP-Ratchet	1552	**854**
	Triplet	N/A	N/A

Experimental results comparing the performance of the RF supertree method to MRP and triplet supertree methods. We used five different supertree analyses: RF supertrees using our SPR local search algorithm with a ratchet starting from either random addition sequence trees (RF-ratchet) or MRP trees (RF-MRP), MRP with TBR branch swapping with (MRP-ratchet) and without (MRP-TBR) a ratchet search, and triplet supertrees with a TBR local search (Triplet). We measured the RF distance to the collection of input trees (RF-distance) and the parsimony score of a best found supertree based on the matrix representation of the input trees. The best RF distance and parsimony scores are in bold.

starting trees. Next, we performed 20 replicates of a parsimony ratchet search with TBR branch swapping. Based on the results of trial analyses, each ratchet search consisted of 25 iterations, each reweighting 15% of the characters. The PAUP* command block for the parsimony ratchet searches was generated using PAUPRat [45]. For each data set, we performed 20 replicates of a TBR local search heuristic starting with random addition sequence trees. Triplet supertrees were constructed using the program from Lin et al. [38]. We were unable to perform ratchet searches with the existing triplet supertree software, and also, due to memory limitations, we were unable to perform triplet supertree analyses on the legume data set.

Our analyses demonstrate the effectiveness of our local search heuristics for the RF supertree problem. In all four data sets, RF-ratchet searches found the supertrees with the lowest total RF distance to the input trees (Table 1). MRP also generally performs well, finding supertrees with RF distances between 0.14% (placental mammals) and 3.3% (sea birds) higher than the best score found by the RF supertree heuristics (Table 1). The triplet supertree method performs as well as the RF supertree method on the small sea bird data set; however, the triplet supertrees for the marsupial and placental mammal data sets have a much higher RF distance to the input trees than either the RF or MRP supertrees (Table 1). For all the data sets, the MRP supertrees had the lowest (best) parsimony score based on a binary matrix representation of the input trees (Table 1). Thus, not surprisingly, it appears that optimizing based on the parsimony score or the triplet distance to the input trees does not optimize the similarity of the supertrees to the input trees based on the RF distance metric (see also [11,13]).

All of the data sets used in this analysis are from published studies that used MRP. Therefore, it is not surprising that MRP performed well (but see [46]). Still, our results demonstrate that MRP leaves some room for improvement. If the goal is to find the supertrees that are most similar to the collection of input trees, the RF searches ultimately provide better estimates than MRP (Table 1).

Interestingly, while the MRP trees tend to have relatively low RF-distance scores, in some cases, such as the legume data set, trees with low RF-distance scores have high parsimony scores (Table 1). Thus, parsimony scores are not necessarily indicative of RF score, and MRP and RF supertree optimality criteria are certainly not equivalent. Still, MRP trees appear to be useful as starting points for RF supertree heuristics. Indeed, in three of the four data sets, the best RF trees were found in ratchet searches beginning from MRP trees (Table 1).

Our program for computing RF supertrees is freely available (for Windows, Linux, and Mac OS X) at http://genome.cs.iastate.edu/CBL/RFsupertrees

Discussion and Conclusion

There is a growing interest in using supertrees for large-scale evolutionary and ecological analyses. Yet there are many concerns about the performance of existing supertree methods, and the great majority of published supertree analyses have relied on only MRP [47]. Since the goal of a supertree analysis is to synthesize the phylogenetic data from a collection of input trees, it makes sense that an effective supertree method should directly seek the supertree that is most similar to the input trees. Our new algorithms make it possible, for the first time, to estimate large supertrees by directly optimizing the RF distance from the supertree to the input trees.

There are numerous alternate metrics to compare phylogenetic trees besides the RF distance, and any of these can be used for supertree methods (see, for example, [11]). Triplet distance supertrees [11,48], quartet-fit and quartet joining supertrees [11,24], maximum splits-fit supertrees [11], and most similar supertrees [49] are all, like RF supertrees, estimated by comparing input trees to the supertree using tree distance measures. All of these methods may provide different, and perhaps equally valid, perspectives on supertree accuracy. Based on our experimental analyses using the RF and triplet supertree method, optimizing the supertree based on different distance measures can result in very different supertrees (Table 1). In the future, it will be important to characterize and compare the performance of these methods in more detail (see, for example, [11,50]).

The results also suggest several future directions for research. Although heuristics guided by local search problems, especially SPR and TBR, have been very effective for many intrinsically difficult phylogenetic inference problems, our experiments indicate that the tree space for RF supertrees is complex. The ratchet approach and also starting from MRP trees appears to improve the performance in the four examples we tested (Table 1). However, more work is needed to identify the most efficient ways to implement our fast local search heuristics. Also, the use of alternative supertrees methods (other than MRP) to generate starting trees might result in a better global strategy to compute RF supertrees and this should be investigated further. We note that the ideas presented in [51] can be directly used to perform efficient NNI-based local searches for the RF supertree problem. In particular, we can show that heuristic searches for the RF supertree problem, which perform a total of p local search steps based on 1, 2, or 3-NNI neighborhoods (see [51]), can all be executed in $O(kn(n + p))$ time; yielding speed-ups of $\Theta(\min\{n, p\})$, $\Theta(n\cdot\min\{n, p\})$ and $\Theta(n^2\cdot\min$

{*n*, *p*}) for heuristic searches that are based on naïve algorithms for 1, 2 and 3-NNI local searches respectively. It would also be interesting to see if heuristics based on TBR perform significantly better than those based on SPR in inferring RF supertrees.

In some cases it might be desirable to remove the restriction that the supertree be binary. In the consensus setting, such a median tree can be obtained within polynomial time [27]; however, finding a median RF tree in the supertree setting is NP-hard [52]. One simple way to estimate a non-binary median tree could be to first compute an RF supertree and then to iteratively (and perhaps greedily) contract those edges in the supertree that result in a reduction in the total RF distance. Thus, our algorithms may even be useful for roughly estimating majority-rule(-) supertrees [28], which are essentially the strict consensus of all optimal, not necessarily binary, median RF trees, and have several desirable properties [29]. These majority-rule(-) supertrees are also the strict consensus of all maximum-likelihood supertrees [53]. Also, there are several alternate forms of the RF distance metric that could be incorporated into our local search algorithms. For example, in order to account for biases associated with the different sizes of input trees, we could normalize the RF distance for each input tree, dividing the observed RF distance by the maximum possible RF distance based on the tree size. Similarly, we could incorporate either branch length data or phylogenetic support scores (bootstrap values or posterior probabilities) from the input trees into the RF distance in order to give more weight to partitions that are strongly supported or separated by long branches (e.g., [25,54]). Our current implementation essentially treats all branch lengths as one and all partitions as equal. The addition of branch length or support data may further improve the accuracy of the RF supertree method.

Acknowledgements
We thank Harris Lin for providing software for the triplet supertree analyses. This work was supported in part by NESCent and by NSF grants DEB-0334832 and DEB-0829674. MSB was supported in part by a postdoctoral fellowship from the Edmond J. Safra Bioinformatics program at Tel-Aviv university.

Author details
[1]The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel. [2]Department of Computer Science, Iowa State University, Ames, IA 50011, USA. [3]Department of Biology, University of Florida, Gainesville, FL 32611, USA.

Authors' contributions
MSB was responsible for algorithm design and program implementation, contributed to the experimental evaluation, and wrote major parts of the manuscript. JGB performed the experimental evaluation and the analysis of the results, and contributed to the writing of the manuscript. OE and DFB supervised the project and contributed to the writing of the manuscript. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 27 June 2009 Accepted: 24 February 2010
Published: 24 February 2010

References
1. Davies TJ, Barraclough TG, Chase MW, Soltis PS, Soltis DE, Savolainen V: **Darwin's abominable mystery: Insights from a supertree of the angiosperms.** *Proceedings of the National Academy of Sciences of the United States of America* 2004, **101(7)**:1904-1909.
2. Bininda-Emonds ORP, Cardillo M, Jones KE, Macphee RDE, Beck RMD, Grenyer R, Price SA, Vos RA, Gittleman JL, Purvis A: **The delayed rise of present-day mammals.** *Nature* 2007, **446(7135)**:507-512.
3. Daubin V, Gouy M, Perriere G: **A Phylogenomic Approach to Bacterial Phylogeny: Evidence of a Core of Genes Sharing a Common History.** *Genome Res* 2002, **12(7)**:1080-1090.
4. Burleigh JG, Driskell AC, Sanderson MJ: **Supertree bootstrapping methods for assessing phylogenetic variation among genes in genome-scale data sets.** *Systematic Biology* 2006, **55**:426-440.
5. Pisani D, Cotton JA, McInerney JO: **Supertrees disentangle the chimerical origin of eukaryotic genomes.** *Mol Biol Evol* 2007, **24(8)**:1752-1760.
6. Webb CO, Ackerly DD, McPeek M, Donoghue MJ: **Phylogenies and community ecology.** *Ann Rev Ecol Syst* 2002, **33**:475-505.
7. Davies TJ, Fritz SA, Grenyer R, Orme CDL, Bielby J, Bininda-Emonds ORP, Cardillo M, Jones KE, Gittleman JL, Mace GM, Purvis A: **Phylogenetic trees and the future of mammalian biodiversity.** *Proceedings of the National Academy of Sciences* 2008, **105(Supplement 1)**:11556-11563.
8. Purvis A: **A modification to Baum and Ragan's method for combining phylogenetic trees.** *Systematic Biology* 1995, **44**:251-255.
9. Pisani D, Wilkinson M: **Matrix Representation with Parsimony, Taxonomic Congruence, and Total Evidence.** *Systematic Biology* 2002, **51**:151-155.
10. Bininda-Emonds ORP, Gittleman JL, Steel MA: **The (super) tree of life: procedures, problems, and prospects.** *Annual Review of Ecology and Systematics* 2002, **33**:265-289.
11. Wilkinson M, Cotton JA, Creevey C, Eulenstein O, Harris SR, Lapointe FJ, Levasseur C, McInerney JO, Pisani D, Thorley JL: **The shape of supertrees to come: Tree shape related properties of fourteen supertree methods.** *Syst Biol* 2005, **54**:419-432.
12. Goloboff PA: **Minority rule supertrees? MRP, Compatibility, and Minimum Flip may display the least frequent groups.** *Cladistics* 2005, **21(3)**:282-294.
13. Wilkinson M, Cotton JA, Lapointe FJ, Pisani D: **Properties of Supertree Methods in the Consensus Setting.** *Syst Biol* 2007, **56(2)**:330-337.
14. Day WH, McMorris F, Wilkinson M: **Explosions and hot spots in supertree methods.** *Journal of Theoretical Biology* 2008, **253(2)**:345-348.
15. Baum BR: **Combining Trees as a Way of Combining Data Sets for Phylogenetic Inference, and the Desirability of Combining Gene Trees.** *Taxon* 1992, **41**:3-10.
16. Ragan MA: **Phylogenetic inference based on matrix representation of trees.** *Molecular Phylogenetics and Evolution* 1992, **1**:53-58.
17. Goloboff PA: **Analyzing Large Data Sets in Reasonable Times: Solutions for Composite Optima.** *Cladistics* 1999, **15(4)**:415-428.
18. Swofford DL: **PAUP*: Phylogenetic analysis using parsimony (*and other methods), Version 4.0b10.** 2002.
19. Roshan U, Moret BME, Warnow T, Williams TL: **Rec-I-DCM3: A Fast Algorithmic Technique for Reconstructing Large Phylogenetic Trees.** *CSB* 2004, 98-109.
20. Bininda-Emonds O, Sanderson M: **Assessment of the accuracy of matrix representation with parsimony analysis supertree construction.** *Systematic Biology* 2001, **50**:565-579.
21. Eulenstein O, Chen D, Burleigh JG, Fernández-Baca D, Sanderson MJ: **Performance of Flip Supertree Construction with a Heuristic Algorithm.** *Systematic Biology* 2003, **53**:299-308.
22. Chen D, Eulenstein O, Fernández-Baca D, Burleigh JG: **Improved Heuristics for Minimum-Flip Supertree Construction.** *Evolutionary Bioinformatics* 2006, **2**.
23. Creevey CJ, McInerney JO: **Clann: investigating phylogenetic information through supertree analyses.** *Bioinformatics* 2005, **21(3)**:390-392.
24. Wilkinson M, Cotton JA: **Supertree Methods for Building the Tree of Life: Divide-and-Conquer Approaches to Large Phylogenetic Problems.**

Reconstructing the Tree of Life: Taxonomy and Systematics of Species Rich Taxa CRC PressHodkinson TR, Parnell JAN 2007, 61-76.
25. Robinson DF, Foulds LR: **Comparison of phylogenetic trees.** *Mathematical Biosciences* 1981, **53(1-2)**:131-147.
26. McMorris FR, Steel MA: **The complexity of the median procedure for binary trees.** *Proceedings of the International Federation of Classification Societies* 1993.
27. Barthélemy JP, McMorris FR: **The median procedure for n-trees.** *Journal of Classification* 1986, **3**:329-334.
28. Cotton JA, Wilkinson M: **Majority-Rule Supertrees.** *Systematic Biology* 2007, **56**:445-452.
29. Dong J, Fernández-Baca D: **Properties of Majority-Rule Supertrees.** *Syst Biol* 2009, **58(3)**:360-367.
30. Day WHE: **Optimal algorithms for comparing trees with labeled leaves.** *Journal of Classification* 1985, **2**:7-28.
31. Pattengale ND, Gottlieb EJ, Moret BME: **Efficiently Computing the Robinson-Foulds Metric.** *Journal of Computational Biology* 2007, **14(6)**:724-735, [PMID: 17691890].
32. Semple C, Steel M: *Phylogenetics* Oxford University Press 2003.
33. Allen BL, Steel M: **Subtree transfer operations and their induced metrics on evolutionary trees.** *Annals of Combinatorics* 2001, **5**:1-13.
34. Swofford DL, Olsen GJ, Waddel PJ, Hillis DM: **Phylogenetic inference.** *Molecular Systematics* Sunderland, Mass: Sinauer AssocHillis DM, Moritz C, Mable BK 1996, 407-509.
35. Bordewich M, Semple C: **On the computational complexity of the rooted subtree prune and regraft distance.** *Annals of Combinatorics* 2004, **8**:409-423.
36. Bansal MS, Burleigh JG, Eulenstein O, Wehe A: **Heuristics for the Gene-Duplication Problem: A Θ(n) Speed-Up for the Local Search.** *RECOMB, Volume 4453 of Lecture Notes in Computer Science* SpringerSpeed TP, Huang H 2007, 238-252.
37. Bansal MS, Eulenstein O: **An Ω(n^2/*log n*) Speed-Up of TBR Heuristics for the Gene-Duplication Problem.** *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2008, **5(4)**:514-524.
38. Lin H, Burleigh JG, Eulenstein O: **Triplet supertree heuristics for the tree of life.** *BMC Bioinformatics* 2009, **10(Suppl 1)**:S8.
39. Bender MA, Farach-Colton M: **The LCA Problem Revisited.** *LATIN, Volume 1776 of Lecture Notes in Computer Science* SpringerGonnet GH, Panario D, Viola A 2000, 88-94.
40. Kennedy M, Page R: **Seabird supertrees: combining partial estimates of procellariiform phylogeny.** *The Auk* 2002, **119**:88-108.
41. Cardillo M, Bininda-Emonds ORP, Boakes E, Purvis A: **A species-level phylogenetic supertree of marsupials.** *Journal of Zoology* 2004, **264**:11-31.
42. Beck R, Bininda-Emonds O, Cardillo M, Liu FG, Purvis A: **A higher-level MRP supertree of placental mammals.** *BMC Evolutionary Biology* 2006, **6**:93.
43. Wojciechowski M, Sanderson M, Steele K, Liston A: **Molecular phylogeny of the "Temperate Herbaceous Tribes" of Papilionoid legumes: a supertree approach.** *Advances in Legume Systematics* Kew: Royal Botanic GardensHerendeen P, Bruneau A 2000, **9**:277-298.
44. Nixon KC: **The parsimony ratchet: a new method for rapid parsimony analysis.** *Cladistics* 1999, **15**:407-414.
45. Sikes DS, Lewis PO: **PAUPRat: PAUP* implementation of the parsimony ratchet.** 2001.
46. Wilkinson M, Pisani D, Cotton JA, Corfe I: **Measuring Support and Finding Unsupported Relationships in Supertrees.** *Syst Biol* 2005, **54(5)**:823-831.
47. Bininda-Emonds OR (Ed): *Phylogenetic supertrees* Springer Verlag 2004.
48. Snir S, Rao S: **Using Max Cut to Enhance Rooted Trees Consistency.** *IEEE/ACM Trans. Comput. Biology Bioinform* 2006, **3(4)**:323-333.
49. Creevey CJ, Fitzpatrick DA, Gayle K Philip RJK, O'Connell MJ, Pentony MM, Travers SA, Wilkinson M, McInerney JO: **Does a tree-like phylogeny only exist at the tips in the prokaryotes?.** *Proc Biol Sci* 2004, **271(1557)**:2551-2558.
50. Thorley JL, Wilkinson M: **A View of Supertree Methods.** *Bioconsensus, Volume 61 of DIMACS: Series in Discrete Mathematics and Theoretic Computer Science, Providence, Rhode Island, USA: American Mathematical Society* 2003, 185-193.
51. Bansal MS, Eulenstein O, Wehe A: **The Gene-Duplication Problem: Near-Linear Time Algorithms for NNI-Based Local Searches.** *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2009, **6(2)**:221-231.
52. Bryant D: **Building trees, hunting for trees, and comparing trees: Theory and methods in phylogenetic analysis.** *PhD thesis* Dept. of Mathematics, University of Canterbury 1997.
53. Steel M, Rodrigo A: **Maximum likelihood supertrees.** *Syst. Biol* 2008, **57**:243-250.
54. Kuhner MK, Felsenstein J: **A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates [published erratum appears in Mol Biol Evol 1995 May;12(3):525].** *Mol Biol Evol* 1994, **11(3)**:459-468.

doi:10.1186/1748-7188-5-18
Cite this article as: Bansal *et al.*: Robinson-Foulds Supertrees. *Algorithms for Molecular Biology* 2010 **5**:18.