LARGE SCALE PHYLOGENOMIC ESTIMATION

BY

PRANJAL VACHASPATI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Abstract

Acknowledgments

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ILS	Incomplete Lineage Sorting

Part I

Introduction

Chapter 1

Phylogenomic estimation

1.1 Organization of this work

Chapter 2 provides much of the background necessary to understand this research into phylogenomic methods, including the relatively minimal amount of biology needed to describe the relevant mathematical models of evolution.

Chapter 2

Background

2.1 Phylogenetic trees

The use of a phylogenetic tree to describe evolutionary relationships was popularized by Darwin, who used a diagram of a tree as the sole illustration in *On the Origin of Species* [30]. While the methods used to estimate these trees have changed drastically, the basic structure and meaning of these trees is more or less the same.

The leaves of a phylogenetic tree represent extant taxa that have been sampled for the analysis, referred to as the taxon set.

Internal nodes represent speciation events. An internal node also represents a species that is the most recent common ancestor of all the descendants of that node, and in some cases (for example, when fossil data is used), may correspond to a known species.

Edges represent the evolution of a species without any speciation events that led to multiple extant species in the dataset (that is to say, speciation events may have occurred along an edge, but only one of those species survived to the present day and was included in the analysis). Edges may have a length, which represents the amount of time between speciation events.

Each edge also corresponds to a split, or bipartition, in the taxon set. The descendants of an internal node n are a clade, and the parent edge of that node separates that clade from the rest of the taxa, which do not have n as an ancestor.

Phylogenetic trees may be rooted or unrooted. In many models of evolution, the root is not identifiable; in other words, every rooting of the same unrooted tree will produce the same distribution of site patterns on the leaves of the tree. Accurately identifying the root of an unrooted tree can be challenging in some cases, especially for gene trees that evolve under ILS or HGT (see Section 2.4).

Trees may be binary or multifurcating (non-binary). Each internal node in a binary tree has degree 3 (except for the root, which has degree 2), while internal nodes in multifurcating trees may have higher degrees. Since speciation events are in reality binary, multifurcations in trees represent an uncertainty as to the ordering of two or more speciation events.

2.2 A simple model of evolution

2.3 The phylogenomic pipeline

A phylogenomic analysis starts with biological specimens. Biologists extract DNA from each specimen and run it through a DNA sequencing machine. The sequencing process will chop the genome into overlapping reads that may range from tens of base pairs to thousands of base pairs, depending on the sequencing technology. An assembler then combines the reads into genomes. The next step is to align the genomes. An alignment algorithm adds "indels" to each genome to make them all the same length. The i^{th} site in each genome is assumed to correspond to the same site in the common ancestor of the organisms being studied, having evolved through a sequence of substitutions, insertions, and deletions. At this point, a variety of methods can be used to analyze the genomes to generate an evolutionary tree.

This dissertation focuses on the last stage of this pipeline: starting from an alignment and generating an evolutionary tree. Nevertheless, the earlier stages are important as errors produced there will result in inaccuracies in the final tree.

The following sections will discuss various factors that make phylogenomic estimation more complicated than the simple model from Section 2.2 would imply.

2.4 Gene tree heterogeneity

A variety of biological effects complicate evolutionary models and create a need for more sophisticated phylogenomic algorithms. Some of the most interesting and important effects result in gene tree heterogenity, where different parts of the genome appear to have evolved in different ways.

subsectionHorizontal gene transfer (HGT)

The easiest to understand cause of gene tree heterogeneity is horizontal (or lateral) gene transfer (HGT or LGT).

2.4.1 Incomplete lineage sorting (ILS)

2.4.2 Short loci

2.5 Additional sources of error in phylogenomic analyses

2.5.1 Rate heterogeneity

2.5.2 Alignment error

2.6 Techniques for phylogenomic estimation

2.6.1 Concatenated maximum-likelihood (CA-ML) analyses

2.6.2 Bayesian Markov chain Monte Carlo methods

2.6.3 Coalescent-aware methods

ASTRAL

MP-EST

NJst

2.7 Supertree estimation

2.7.1 Methods

MRL

2.7.2 Divide and conquer methods

Part II

Supertree Estimation

Chapter 3

FastRFS

3.1 Introduction

Supertree estimation is the problem of computing a tree on a set S of taxa from a set of estimated trees (called “source trees”) on subsets of S. Traditionally, the purpose of supertree estimation was to combine published species trees estimated by different research groups around the world, using different datasets and different methods. Supertree methods have been used to construct many species trees, and the development of supertree methods is an area of very active research (see [17] for some of the early literature, and [3, 105, 134] for some more recent methods).

More recently, supertree estimation has been used within divide-and-conquer frameworks, in which a large and potentially heterogeneous dataset is divided into overlapping smaller subsets, trees are estimated on each subset, and then combined into a tree on the full dataset using a supertree method. These divide-and-conquer methods thus enable the application of statistical phylogeny estimation methods to scale to larger datasets [14, 52, 104, 145]. Each of these methods has been able to improve the accuracy and/or speed of its base method. Thus, supertree computation provides an essential tool for both moderate- and large-scale phylogeny estimation, and is relevant to both gene tree estimation and species tree estimation.

One of the popular approaches to supertree estimation is the NP-hard Robinson-Foulds Supertree problem [9], which seeks a binary tree that has the minimum total Robinson-Foulds [112] distance to the input source trees. The best known local search heuristic for the Robinson-Foulds Supertree is MulRF [24], but PluMiST [63] is a new method that shows promise; to our knowledge, there are no other methods that are competitive with these two.

One of the exciting properties of the Robinson-Foulds Supertree problem is that it is closely related to the Maximum Likelihood Supertree problem, which seeks a supertree that is the most likely to have produced the observed source trees under a simple exponential model of phylogenetic error [131]. Although the two problems are not identical (as established in [19]), it seems likely that good solutions to the Robinson-Foulds Supertree problem will be good solutions to the Maximum Likelihood Supertree problem. However, the only technique for the Maximum Likelihood Supertree problem that we are aware of, L.U.-st [3], is very computationally intensive, making it infeasible for use on biological datasets [2].

In this paper, we report on a new method, FastRFS (Fast Robinson-Foulds Supertrees) for finding optimal Robinson-Foulds Supertrees in a constrained search space. Unlike the previous methods for Robinson-Foulds Supertrees, which depended on heuristic searches through tree space, the method we have designed uses dynamic programming (DP) to find an exact solution to the Robinson-Foulds Supertree problem within a constrained search space.

This algorithmic strategy of using dynamic programming to find a species tree optimizing some criterion within a constrained search space was first used in [45]; since that time, the approach has been used in other phylogenetic estimation methods [11, 20, 94, 95, 139, 155]. Most of these methods constrain the search space for their optimization problem by computing a set X of allowed bipartitions (i.e., splits of the leafset into two parts, each defined by deleting edges in the species tree that will be constructed) from the input, and require that the output tree draw its bipartitions from X. These methods run in time that is polynomial in the number of species, source trees, and $|X|$. Many of these methods specify X to be the set of bipartitions in the input source trees, but expanding the set can improve accuracy [94].

The supertree method we present, FastRFS, is a combination of the polynomial time dynamic programming algorithm for the constrained Robinson-Foulds Supertree problem we have developed and the technique we use to define the set X from the input source trees. The basic FastRFS method uses ASTRAL-2 to define the set X of allowed bipartitions from the input set of source trees. We also explore an enhanced version where we add additional bipartitions (beyond those computed by ASTRAL-2) to the set X defined by ASTRAL-2. We define the additional bipartitions by computing fast supertrees on the input set, and then add their bipartitions to X; this approach ensures that we find RFS criterion scores that are at least as good as the trees we use to define the set X of allowed bipartitions, and also at least as good as the trees obtained by the basic FastRFS method. By only adding bipartitions from supertrees that we can compute quickly, the enhanced FastRFS method is able to complete quickly, and provides improved criterion scores.

We evaluate these two versions of FastRFS in comparison to leading methods for supertree estimation on a collection of biological and simulated datasets with 100 to 2228 species that were used in prior publications to evaluate supertree methods [105, 133, 134]. We compare FastRFS to PluMiST, the current best performing method (in terms of criterion scores) for the Robinson-Foulds Supertree problem, and also to MulRF, the most well known software for this optimization problem. We also compare FastRFS to Matrix Representation with Likelihood (MRL) [105], ASTRID [141], and ASTRAL-2 [94]. MRL is the maximum likelihood counterpart to the well known Matrix Representation with Parsimony (MRP) method, and has produced topologically more accurate supertrees than leading MRP heuristics [105]. ASTRID and ASTRAL-2 are methods for species tree estimation that take gene tree heterogeneity arising from incomplete lineage sorting into account, and have had good accuracy on large phylogenomic datasets. We evaluate these methods with respect to RFS criterion scores (which can be evaluated on both simulated and biological datasets), topological accuracy in estimating the true supertree (which can only be evaluated on simulated datasets), and wall clock running time.

3.2 Materials and methods

Every model tree and estimated supertree in this study is a fully resolved tree, and no two leaves have the same label; the source trees are unrooted trees with leaves drawn from (possibly proper) subsets of the full set of taxa, and may contain polytomies (nodes of degree greater than three). We let $T|Q$ denote the tree obtained by restricting the tree T to the subset Q of its leafset, and then suppressing nodes of degree two. We let $\mathcal{L}(T)$ denote the leafset of a tree T. The deletion of an edge e from a tree T induces a bipartition of $\mathcal{L}(T)$ into two sets A and B, denoted by $[A, B]$. Every unrooted tree T is defined by its set $Bip(T)$ of bipartitions. The Robinson-Foulds (RF) distance between trees T and T' that are on the same leafset is the number of bipartitions that are in one tree but not the other (i.e., $RF(T, T') = |Bip(T) \triangle Bip(T')|$). Note that when T and T' have the same leafset, then $RF(T, T') = 0$ if and only if $T = T'$.

We extend the definition of RF distance to trees t and T with nested leafsets (i.e., $\mathcal{L}(t) \subseteq \mathcal{L}(T)$) to be the RF distance between $T|\mathcal{L}(t)$ and t, and denote this distance by $RF(T, t)$. Given a set $\mathcal{T}$ of trees and tree T satisfying $\mathcal{L}(t) \subseteq \mathcal{L}(T)$ for all $t \in \mathcal{T}$, we define $RF(T, \mathcal{T}) = \sum_{t \in \mathcal{T}} RF(T, t)$. A binary tree T with leafset $S = \cup_{t \in \mathcal{T}} \mathcal{L}(t)$ that minimizes $RF(T, \mathcal{T})$ is the Robinson-Foulds Supertree for $\mathcal{T}$, and is denoted T_{RFS}.

Finding a Robinson-Foulds Supertree is NP-hard; however, the Constrained Robinson-Foulds Supertree Problem constrains the search space using a set X of allowed bipartitions, and can be solved in polynomial time, as we will show.

Constrained Robinson-Foulds Supertree Problem:

- Input: Set $\mathcal{T}$ of trees and set X of bipartitions of the taxon set S, where $S = \bigcup_{t \in \mathcal{T}} \mathcal{L}(t)$.
- Output: Unrooted binary tree $T_{RFS(c)}$ that minimizes $RF(T, \mathcal{T})$, subject to the constraint that every bipartition in $T_{RFS(c)}$ is drawn from X.

The Dynamic Programming Algorithm to solve Constrained Robinson-Foulds Supertrees. While the Robinson-Foulds Supertree problem is stated in terms of minimizing the total Robinson-Foulds distance to the source trees, we will rephrase it as *maximizing the bipartition support* from the source trees. This formulation will make it easy for us to present and explain the dynamic programming approach we have developed.

Let t be a source tree with leafset S' and let T be a tree with leafset Y, so that $S' \subseteq Y \subseteq S$. Let $[A', B']$ be a bipartition in t. We will say that $[A', B']$ supports T if there is a bipartition $[A, B]$ in T such that $A' = S' \cap A$ and $B' = S' \cap B$. We will also say that the bipartition support of t for T is the number of bipartitions in t that support T, and that the bipartition support from $\mathcal{T}$ for T is the bipartition support for T from all the trees in $\mathcal{T}$.

Observation 1. For any set $\mathcal{T}$ of source trees, a binary tree T with leafset $S = \cup_{t \in \mathcal{T}} \mathcal{L}(t)$ that has the maximum bipartition support from $\mathcal{T}$ is an optimal solution to the Robinson-Foulds Supertree problem.

Recall that the input includes a set X of allowed bipartitions. A clade in a rooted tree is a set of leaves that constitute all the leaves below some selected node in the rooted tree. We define a set $\mathcal{C}$ of allowed clades, by setting $\mathcal{C} = \{A : \exists [A, B] \in X\}$ (i.e., $\mathcal{C}$ contains every half of every bipartition in X).

Let t be an unrooted tree with leafset S', let T be a rooted binary tree with leafset Y where $S' \subseteq Y$, and let $[A', B']$ be a bipartition in t. We will say that $[A', B']$ supports T if $T|S'$ contains A' or B' (or both) as clades. We define the bipartition support of source tree $t \in \mathcal{T}$ for the rooted tree T to be the number of bipartitions in t that support T, and the bipartition support of $\mathcal{T}$ for T to be the total of the bipartition support from all the source trees in $\mathcal{T}$ for T. Furthermore, given node v in T, we let T_v denote the subtree of T rooted at v; note that every node in T_v is also a node in T.

Observation 2. For all sets $\mathcal{T}$ of source trees and all rooted trees T with leafset $S = \cup_{t \in \mathcal{T}} \mathcal{L}(t)$, the bipartition support of $\mathcal{T}$ for T is the same as the bipartition support of $\mathcal{T}$ for the unrooted version of T.

By Observation 2, we can solve the Constrained Robinson-Foulds Supertree problem by finding a rooted tree with leafset S that has the maximum bipartition support, and then unrooting this tree.

For the rest of this discussion, T will denote a rooted binary tree with leafset $Y \subseteq S$, with all its clades drawn from $\mathcal{C}$. We will show that we can write the bipartition support for T from a source tree t as the sum of the bipartition support for the clades in T, which will allow us to construct a dynamic programming algorithm.

Consider an internal node v in T, and let v_1 and v_2 be its two children. Let the clade below v be A, the clade below v_1 be A_1, and the clade below v_2 be A_2. Deleting v from T splits Y into three parts: A_1, A_2 and $A_3 = Y \setminus A$. We will describe this by saying v defines the ordered tripartition (A_1, A_2, A_3), with the understanding that (A_1, A_2, A_3) and (A_2, A_1, A_3) are equivalent, and both correspond to node v. Note that if $Y \neq S$, then the tripartition defined by v will not cover all the elements of S. Also, we will require that A_1 and A_2 be allowed clades (i.e., in $\mathcal{C}$), but we make no such constraint on A_3.

Suppose that source tree t with leafset S' has a bipartition $[U', V']$ that supports T; thus, $T|S'$ must have U' or V' (or both) as clades. We wish to associate this bipartition to exactly one node in T, so that we can compute the bipartition support without having to correct for over-counting, and so that the dynamic programming algorithm is simple.

Case 1: $T|S'$ contains only one of these two clades. Suppose $T|S'$ contains U' but not V' as a clade. If $T|S'$ does not contain any leaves from V', we do *not* assign $[U', V']$ to any node in T. If $T'|S'$ does contain at least one leaf from V', we follow the path from the MRCA of U' towards the root until we reach the first node w that has at least one element of V' in the subtree below it, and we assign $[U', V']$ to w.

Case 2: $T|S'$ contains both U' and V' as clades. We assign $[U', V']$ to the MRCA of $U' \cup V'$.

The following lemma follows directly from the description of the assignment process:

Lemma 3. For any bipartition $\pi = [U', V']$ and any tree T, π is assigned to node w in T if and only if w defines a tripartition (A_1, A_2, A_3) where $U' \subseteq A_1, V' \cap A_1 = \emptyset$, and $V' \cap A_2 \neq \emptyset$. If π supports T, then there is a unique node in T satisfying this constraint. However, if no such node exists, π does not support T, and so is not assigned to any node in T.

Lemma 4. Let T be a rooted tree on set Y, and let v be a node in T other than the root. Let $[U', V']$ be a bipartition in a source tree t that supports both T and T_v, and suppose that $[U', V']$ is assigned to node w in T and node w' in T_v. Then $w = w'$.

Proof. By Lemma 3, $[U', V']$ is assigned to the unique node w' in T_v that defines a tripartition (A_1, A_2, A_3) where $U' \subseteq A_1, V' \cap A_1 = \emptyset$, and $V' \cap A_2 \neq \emptyset$. Since T_v is a rooted subtree of T, the node w' exists in T, and defines the tripartition (A_1, A_2, A_3') that differs from the tripartition above only in the third coordinate. By Lemma 3, it follows that $w = w'$. □

Note that the assignment of bipartitions to nodes in trees depends only on the first two components of the tripartition for the node. We make the following definition:

Definition 5. Let A_1, A_2 be a disjoint pair of allowed clades. We define *support*(A_1, A_2) to be the number of bipartitions in the source trees that map to a tripartition (A_1, A_2, Z) for some Z.

Theorem 6. The bipartition support from $\mathcal{T}$ for a rooted binary tree T is

$$\sum_{(A_1,A_2,A_3)\in Trip(T)} \mathit{support}(A_1, A_2),$$

where $Trip(T)$ denotes the set of tripartitions defined by the nodes of T.

Proof. The prior discussion establishes that for a given source tree $t \in \mathcal{T}$ and bipartition $\pi_e \in Bip(T)$ that supports T, there is exactly one tripartition in $Trip(T)$ that π_e is mapped to. Furthermore, if π_e does not support T, then it is not mapped to any tripartition in $Trip(T)$. The theorem follows. □

Theorem 7. Let $\mathcal{T}$ be a set of source trees with S the set of taxa that appear as a leaf in at least one tree in $\mathcal{T}$, and let $\mathcal{C}$ be the set of allowed clades. Set $BPS(\{s\}) = 0$ for all $s \in S$, and let $BPS(A)$ for $A \in \mathcal{C}$ with $|A| \geq 2$ be the maximum bipartition support over all rooted binary trees T on clade A where T draws its clades from $\mathcal{C}$. Then, for $A \in \mathcal{C}, |A| \geq 2$,

$$\begin{aligned} &BPS(A) = \\ &\qquad \max\{BPS(A_1) + BPS(A_2) + \mathit{support}(A_1, A_2) : \\ &\qquad A = A_1 \cup A_2, A_1 \cap A_2 = \emptyset, A_i \in \mathcal{C}\} \end{aligned} \tag{3.1}$$

Proof. Let $A \in \mathcal{C}$ be arbitrary, with $|A| \geq 2$. Let $BPS^*(A)$ denote the maximum achievable bipartition support of any rooted tree on A that draws its clades from $\mathcal{C}$, and let $BPS(A)$ be the value as computed by Equation 3.1. We will prove by induction on the size of A that $BPS^*(A) = BPS(A)$.

The base case is $A = \{a, a'\}$. There is only one rooted tree on A, and it has bipartition support $\mathit{support}(\{a\}, \{a'\})$, which is equal to $BPS(A)$. Hence $BPS^*(A) = BPS(A)$ for $|A| \leq 2$. Now let $|A| > 2$ be arbitrary, and let T be a binary rooted tree with leafset A having the largest bipartition support from the trees in $\mathcal{T}$, and drawing its clades from $\mathcal{C}$. The inductive hypothesis is that $BPS(A') = BPS^*(A')$ for all proper subsets A' of A where $A' \in \mathcal{C}$.

Let v_1 and v_2 be the two children of the root of T, A_1 and A_2 be the leafsets of the subtrees of T rooted at v_1 and v_2, and T_1 and T_2 be the subtrees of T rooted at v_1 and v_2, respectively. By the inductive hypothesis, $BPS(A_1) = BPS^*(A_1)$ and $BPS(A_2) = BPS^*(A_2)$. Because T optimizes the bipartition support of all rooted binary trees on A given the constraint set, T_1 and T_2 have the highest bipartition support of all rooted binary trees on A_1 and A_2, respectively, given the constraint set. By Theorem 1, the bipartition support of T_i is the sum of *support*(X, Y) for all tripartitions defined by the nodes of T_i, for $i = 1, 2$, and the bipartition support of T is the sum of *support*(X, Y) for all tripartitions defined by the nodes of T. Hence, the bipartition support of T is $BPS(A_1) + BPS(A_2) + \mathit{support}(A_1, A_2)$. Thus, $BPS^*(A) = BPS(A_1) + BPS(A_2) + \mathit{support}(A_1, A_2)$, and so $BPS^*(A) \leq BPS(A)$.

To complete the proof, we need only show that $BPS(A) \leq BPS^*(A)$. So suppose $BPS(A) > BPS^*(A)$. Then there is a bipartition $[A'_1, A'_2]$ of A such that $BPS(A'_1) + BPS(A'_2) + \mathit{support}(A'_1, A'_2) > BPS(A_1) + BPS(A_2) + \mathit{support}(A_1, A_2)$. Let T'_1 and T'_2 be the rooted trees on A'_1 and A'_2 having quartet support $BPS^*(A'_1)$ and $BPS^*(A'_2)$, respectively, with clades drawn from $\mathcal{C}$, and let T' be the binary rooted tree on A with subtrees T'_1 and T'_2. Then T' draws its clades from $\mathcal{C}$ and has bipartition support that is strictly greater than that of T. This contradicts the assumption that T had the largest bipartition support among all rooted binary trees drawing its clades from $\mathcal{C}$. Hence, $BPS(A) \leq BPS^*(A)$. We have shown that $BPS(A) \leq BPS^*(A)$ and $BPS^*(A) \leq BPS(A)$, and so $BPS(A) = BPS^*(A)$. Since A was arbitrary, the theorem follows. □

The Dynamic Programming Algorithm The input is a pair $(\mathcal{T}, X)$ where $\mathcal{T}$ is a set of source trees and X is a set of allowed bipartitions.

- Preprocessing: Compute the set $\mathcal{C}$ of allowed clades, and order them by cardinality from smallest to largest. Compute the set S of taxa. Set $BPS(\{s\}) = 0$ for all $s \in S$. Compute *support*(A_1, A_2) for every pair of disjoint allowed clades A_1, A_2.

- For each $A \in \mathcal{C}$ with $|A| \geq 2$, in order of size (from smallest to largest), set

$$BPS(A) = \max\{BPS(A_1) + BPS(A_2) + \mathit{support}(A_1, A_2)\},$$

  where A_1 and A_2 are disjoint allowed clades and $A = A_1 \cup A_2$.

- Return $BPS(S)$.

- Compute a rooted binary tree achieving this score using backtracking, and then unroot it to produce a Robinson-Foulds Supertree.

Theorem 8. The dynamic programming algorithm finds an optimal solution to the constrained Robinson-Foulds Supertree problem, and does so in $O(|X|^2nk)$ time, where there are n taxa and k source trees.

Proof. Let $(\mathcal{T}, X)$ (where $\mathcal{T}$ is the set of source trees and X is a set of bipartitions on the species set S) be an input to the constrained Robinson-Foulds supertree problem, and let $\mathcal{C}$ be the set of halves of these bipartitions. By Theorem 2, the dynamic programming algorithm correctly computes the best achievable bipartition support for any rooted supertree drawing its clades from set $\mathcal{C}$. Backtracking produces a rooted T achieving that optimal score, and unrooting T produces T', which has the same optimal score. By construction, T draws its clades from $\mathcal{C}$, and so T' draws its bipartitions from X. Hence, the output from the algorithm, T', is a supertree that draws its bipartitions from X and that achieves the best possible bipartition support score of all supertrees drawing their bipartitions from X; this establishes correctness.

For the running time analysis, we begin with the preprocessing step. Note that $|\mathcal{C}| = 2|X|$ and that there are $O(nk)$ bipartitions in the source trees. For each of the $O(|X|)$ allowed clades A and each half Y_1 of the $O(nk)$ source tree bipartitions, we determine if $Y_1 \subseteq A$; this takes $O(n)$ time per comparison, for a total cost of $O(|X|n^2k)$ time. Once this is done, we can compute *support*(A_1, A_2) for every pair A_1, A_2 of disjoint allowed clades, using $O(|X|^2nk)$ additional time. Since $|X| \geq n - 3$, $|X|n^2k \leq |X|^2nk$; hence, the preprocessing is done in $O(|X|^2nk)$ time. The second phase, where we compute $BPS(A)$ for the allowed clades A, is easily seen to take $O(|X|)$ time per clade, provided that the preprocessing is done first, and the calculations are done in the proper order. Hence, the total time is $O(|X|^2nk)$ time. □

The basic FastRFS method. The input to the FastRFS method is a set $\mathcal{T}$ of unrooted source trees, but they do not need to be binary trees (i.e., polytomies are allowed). In the basic FastRFS method, we use ASTRAL-2 to compute the set X of allowed bipartitions. The technique in ASTRAL-2 for computing the set X of allowed bipartitions produces a set that contains at least one compatible subset of $n-3$ bipartitions, where $n = |S|$; as a result, FastRFS is guaranteed to return a fully resolved tree on every input. See [91] for details on how ASTRAL-2 computes the set X.

FastRFS-enhanced and ASTRAL-enhanced. The enhanced version of FastRFS, which we write as FastRFS-enhanced, operates by computing a set Z of supertrees that can be computed quickly on $\mathcal{T}$, and then adds the bipartitions from trees in Z to the set X that is computed by ASTRAL-2. This approach ensures that the RFS criterion score found by FastRFS-enhanced will be *at least as good* as any tree in Z.

In our study, we used one or both of ASTRID and MRL for our set Z. ASTRID computes a matrix of average pairwise "internode distances" (the number of edges in the path between two species in a tree), and then computes a tree on the distance matrix. When the distance matrix has no missing data, ASTRID uses FastME [36], a fast and accurate method to compute the supertree; however, when the distance matrix has missing entries, it uses BioNJ* [28], a method that is slower and not quite as accurate (see [141] for

Method	100	100	100	100	500	500	500	500	1000	1000	1000	1000
Scaffold %	20	50	75	100	20	50	75	100	20	50	75	100
# Replicates	9	10	10	10	8	10	10	10	10	10	10	10
ASTRAL	32	31	38	45	170	190	225	274	365	414	502	591
ASTRAL-enhanced	32	30	38	45	163	182	221	274	337	393	491	591
ASTRID	40	41	50	41	360	914	905	223	1066	2447	2370	470
MRL	30	30	36	42	158	179	202	223	309	362	412	474
MulRF	32	34	38	**40**	282	315	279	229	–	–	–	–
PluMiST	31	29	**34**	**40**	210	245	246	214	–	–	–	–
FastRFS-basic	**29**	29	**34**	**40**	152	173	191	209	325	366	394	434
FastRFS-enhanced	**29**	**28**	**34**	**40**	**148**	**166**	**186**	**206**	**292**	**347**	**384**	**426**

Table 3.1: Average Robinson-Foulds Supertree criterion scores on the simulated datasets; lower is better. No results shown for the 1000-taxon datasets for MulRF and PluMiST, due to time constraints; otherwise, results are shown for those datasets for which all methods completed. The best result shown for a given model condition is boldfaced.

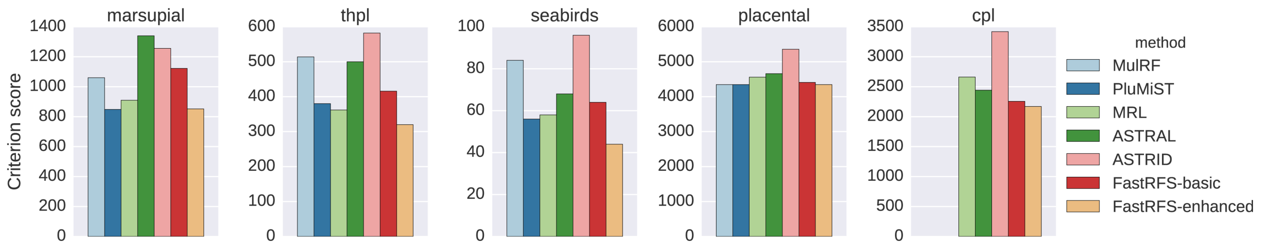


Figure 3.1: RFS criterion scores on biological data of supertree methods; lower is better. MulRF and PluMiST could not be run on the CPL dataset due to its large size; hence no values are shown for those methods on that dataset. Overall, FastRFS-enhanced produces the best RFS criterion scores on these datasets.

a comparison of ASTRID using BioNJ* and ASTRID using FastME). In our experiments, we include the MRL tree in Z, and we also include the ASTRID tree for those inputs where the internode distance matrix has no missing entries. We similarly define ASTRAL-enhanced using the same set of extra trees as for FastRFS-enhanced.

Datasets. We used a collection of published simulated and biological datasets that have been used in other studies [133] to evaluate supertree methods, all of which are available online at `http://www.cs.utexas.edu/users/phylo/datasets/supertrees.html`.

The simulated data, referred to as "SMIDgen" in [133], are generated using a taxon sampling strategy that mimics biological practice. These datasets have 100, 500, or 1000 taxa, with up to 25 source trees per replicate. Each supertree input has several "clade-based" source trees and a "scaffold tree", which are estimated using maximum likelihood heuristics on a concatenation of gene sequence alignments. Some genes are "universal" and so are present in every species; others evolve within the species tree under a birth-death model in which birth happens once but death (i.e., gene disappearance) can happen several times; therefore,

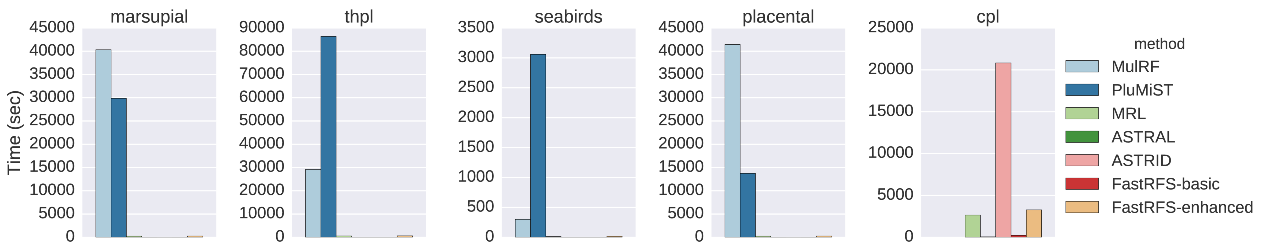


Figure 3.2: Sequential running times (in seconds) on biological data of supertree methods. MulRF and PluMiST could not be run on the CPL dataset, due to its large size; hence no values are shown for those methods on that dataset.

unless the gene is born at the root of the species tree, it will be present only within a clade within the tree. Sequences then evolve down the gene trees under the GTRGAMMA model of site evolution. The scaffold tree is based only on the universal genes, and have a random subset of the species set; the clade-based trees are obtained by selecting a clade in the tree and then a set of genes that covers that clade well. As shown in [133], the density of the scaffold tree (i.e., the percentage of the full set of taxa that are in the scaffold dataset) has a large impact on the topological accuracy of the resultant estimated supertree. These simulated data enable us to evaluate topological accuracy as well as criterion score.

We include the biological datasets that were also studied in [133]: CPL (comprehensive papilinoid legumes) [87], Marsupials [23], Placental Mammals [15], Seabirds [56], and THPL (temperate herbaceous papilionmoid legumes) [152]. These range in size from 116 species (Placental Mammals) to 2228 (CPL), and with as few as 7 source trees (Seabirds) to as many as 726 (Placental Mammals).

Methods. In addition to the two FastRFS variants (basic and enhanced), we computed supertrees using MRL, ASTRID, ASTRAL-2, MulRF, and PluMiST. For MRL, we compute the MRP matrix using "mrpmatrix" available at `github.com/smirarab/mrpmatrix`, and we use RAxML [129] version 8.2.4 under the BINGAMMA model with seed 12345 on the MRP matrix. We ran MulRF version 1.2 [24] and PluMiST version 1.1 [63]. We ran PluMiST in default mode, and we ran MulRF ten times, and report results for the tree with the best criterion score. We ran ASTRAL-2 version 4.7.12 [94] (henceforth referred to as ASTRAL) and ASTRID version 1.1 [141], both in default mode. Each of these methods produces fully resolved unrooted trees.

ASTRAL produces a supertree that minimizes the total quartet distance to the input source trees (equivalently, it produces a supertree that maximizes the total quartet tree support) subject to a constrained set X of bipartitions that it computes from the input source trees.

We tested an enhanced version of ASTRAL (analogous to FastRFS-enhanced), in which we added the bipartitions from MRL and ASTRID to the set X; this enables a direct comparison of FastRFS-enhanced and ASTRAL-enhanced. Although FastRFS-enhanced is guaranteed to find RFS criterion scores that are at least as good as ASTRAL-enhanced, the comparison with respect to tree topology accuracy makes it possible to evaluate the two optimization criteria (minimize quartet distance or minimize Robinson-Foulds distance) and their impact on topological accuracy. Finally, we tested the impact of adding the bipartitions from just one tree (MRL or ASTRID) to the set X on FastRFS, to determine the relative impact of each additional tree.

Measurements. We can use the simulated data to explore performance with respect to criterion scores as well as tree estimation error. However, since there is no known true supertree for the biological datasets, we use the biological datasets to explore performance only with respect to criterion scores.

For tree estimation error (explored only on the simulated datasets), we report the normalized bipartition distance (also called the Robinson-Foulds error rate) between the estimated and true trees. The Robinson-Foulds (RF) error rate is $\frac{RF(T,T')}{2n-6}$, where $RF(T,T')$ is the RF distance between the true tree T and the estimated tree T', and n is the number of leaves in T). Hence, the RF error rate is between 0 and 1, and is equal to 0 if and only if the two trees are identical.

We also report the Robinson-Foulds Supertree criterion score (i.e., the total Robinson-Foulds distance between the estimated supertree and the input source trees) for all datasets; this value is bounded from above by $(2n-6)k$, where n is the total number of species and k is the total number of source trees.

Although the criterion scores and tree error metrics both refer to the Robinson-Foulds distance, the criterion score is based on the RF distance to the input source trees, and the tree error metric refers to the RF distance to the model tree, which is unknown. Hence these are two different ways of evaluating methods.

Most of the methods are sequential codes; however, FastRFS is parallelized to run on 8 cores and we run MulRF 10 times in parallel and take the best tree. We report wall clock running times for all codes; except when the differences are large, comparisons between running times are not reliable. Running times for FastRFS-enhanced include the time to compute the MRL tree and the ASTRID distance matrix, and, if the distance matrix has no missing data, the time to run FastME on the distance matrix (i.e., to fully compute the ASTRID tree).

Experiments. We performed experiments to evaluate the different supertree methods with respect to Robinson-Foulds criterion score, topological accuracy of the supertree, and running time.

Method	100	100	100	100	500	500	500	500	1000	1000	1000	1000
Scaffold %	20	50	75	100	20	50	75	100	20	50	75	100
# Replicates	9	10	10	10	8	10	10	10	10	10	10	10
ASTRAL	**11.7**	14.0	11.6	10.0	15.3	14.8	12.7	11.2	16.9	15.7	13.6	11.6
ASTRAL-enhanced	11.8	**13.1**	11.5	10.0	14.8	14.1	12.6	11.2	**16.3**	**15.1**	13.5	11.6
ASTRID	15.8	18.7	17.1	9.6	26.0	50.1	45.4	**10.5**	35.6	58.1	52.0	**11.2**
MRL	13.6	13.6	11.2	10.8	15.4	14.3	12.1	11.2	17.4	**15.1**	13.5	12.2
MulRF	22.1	26.0	15.3	9.3	46.9	40.3	27.4	12.6	–	–	–	–
PluMiST	25.9	16.6	11.5	9.3	35.4	29.5	22.4	10.9	–	–	–	–
FastRFS-basic	13.5	14.3	**10.5**	**9.1**	14.5	14.3	12.4	11.1	17.3	15.6	13.5	12.0
FastRFS-enhanced	13.5	13.4	10.6	9.3	**14.3**	**13.9**	**12.0**	10.8	16.7	**15.1**	**13.4**	11.8

Table 3.2: Supertree topology estimation error on simulated datasets, measured using the Robinson-Foulds error rate, expressed as a percentage. The best result for each model condition is boldfaced. No results are shown for PluMiST or MulRF on the 1000-taxon simulated datasets due to running time limitations for these methods. Results are averaged over the completed replicates.

3.3 Results and discussion

Impact of the constraint set on criterion scores. Our initial experiment evaluated the impact on the criterion scores found by FastRFS of adding bipartitions from the MRL tree and/or the ASTRID tree to the constraint set. In general, FastRFS with the MRL tree alone added was nearly as good as FastRFS-enhanced (i.e., with both ASTRID and MRL trees added), and FastRFS with MRL found substantially better criterion scores than FastRFS with just the ASTRID tree added. Nevertheless, since adding the ASTRID tree did help occasionally, and since ASTRID is so quick to run when the distance matrix is complete, we continued using it for FastRFS-enhanced. See supplementary materials for these results.

Criterion scores for the simulated datasets. By design, FastRFS-enhanced will always find criterion scores that are at least as good as those found by ASTRAL-enhanced, FastRFS-basic, ASTRAL, and MRL. Hence, the only methods that could possibly find better scores than FastRFS-enhanced are PluMiST, MulRF, and ASTRID. We show the Robinson-Foulds Supertree criterion scores in Table 3.1; note that lower is better. PluMiST failed to complete on three datasets (one replicate in the 100-taxon and two replicates in the 500-taxon datasets, each with 20%-scaffolds); we report results only on the remaining datasets. Both PluMiST and MulRF had very large running times on the 500-taxon datasets; therefore, we did not attempt to run them on the 1000-taxon datasets. All other methods succeeded in completing on all datasets we examined.

FastRFS-enhanced found the best (lowest) Robinson-Foulds Supertree (RFS) criterion scores of all methods for all datasets; FastRFS-basic also found these best scores for three of the four 100-taxon model conditions, but otherwise found higher scores. PluMiST found better RFS criterion scores than MulRF in 7 of the 8 model conditions, and matched in 1 condition. ASTRID had the worst performance of all methods, with much larger criterion scores on all the sparse scaffold model conditions. These are the same conditions in which the internode distance matrix has missing entries, suggesting that the reduced accuracy is largely due to the missing data in the distance matrix.

Certain additional trends are worth noting. First, although PluMiST did well on the 100-taxon datasets, it was not so competitive with FastRFS-enhanced or even FastRFS-basic on the 500-taxon datasets, suggesting that the number of taxa may impact the ability of PluMiST to find trees with good criterion scores. ASTRAL-enhanced matched or improved on the RFS criterion scores compared to ASTRAL; this is interesting because it does not follow from the algorithm design (the two methods seek the tree that minimizes the quartet distance, not the RFS criterion). MRL, although never coming in first, often had very good results, coming just behind FastRFS-basic for overall performance.

Criterion scores on biological datasets. We were unable to run PluMiST and MulRF on the CPL dataset, the largest in our collection, due to its size: at 2228 species, the running time needed for these two methods is excessive. Criterion scores on the biological datasets follow very similar patterns as observed on the simulated datasets (Fig. 3.1). Overall, FastRFS-enhanced had the best criterion scores: the best on four datasets, and close to best on the last dataset (Marsupials). PluMiST tied for best with FastRFS-enhanced on two of the four datasets on which it can run, had the second best score on seabirds, and third best on THPL. Hence, PluMiST is in second place. Interestingly, the dataset on which PluMiST was not able to find one of the top two scores was the second largest dataset, with more than 500 species. Thus, just as we saw on the simulated datasets, the number of species seems to impact the relative performance of PluMiST in comparison to other methods.

The next two best methods are MRL and FastRFS-basic, which had close performance, but MRL was slightly better. ASTRAL and MulRF are next, again with mixed performance (MulRF was better on two datasets and ASTRAL was better on the other two). Finally, ASTRID had the worst performance of all methods - coming in dead last on four of the five datasets. It is worth noting that all but two of the datasets produced distance matrices with missing entries, and ASTRID did better than ASTRAL on one of the two datasets (marsupial) that produced a complete distance matrix.

Topological accuracy. Since the true supertree is not known for the biological datasets, we evaluate topological accuracy only on the simulated datasets red (Table 2). All methods improved in accuracy with the increase in the scaffold density, so that error rates were generally highest for 20%-scaffolds and lowest for 100%-scaffolds. The differences between methods on the 100%-scaffolds were generally small, but there were large differences under the other conditions. ASTRID had very poor accuracy except for those with

100%-scaffolds, and MulRF and PluMiST also had poor accuracy with the lower density scaffolds.

The remaining methods (MRL, the two ASTRAL versions, and the two FastRFS versions) were fairly close in accuracy. However, MRL was never more accurate than FastRFS-enhanced, and was only the top performing method for one model condition (where it tied with FastRFS-enhanced). ASTRAL-enhanced was more accurate than ASTRAL on 8 conditions, tied on 1 condition, and less accurate on 3 conditions. FastRFS-enhanced was more accurate than FastRFS-basic on 9 model conditions, tied on 1 condition, and worse on 2 conditions. FastRFS-enhanced was more accurate than ASTRAL-enhanced on 8 of the 12 model conditions, tied on 1 condition, and worse on 3 conditions.

FastRFS-enhanced was the top performing method on 5 of the 12 model conditions; the next best performing method was ASTRAL-enhanced, which was the top performing method in 3 of the 12 model conditions. Thus, overall FastRFS-enhanced provided the best accuracy of the tested supertree methods. These results, and especially the pairwise comparisons, suggest that optimizing the Robinson-Foulds Supertree criterion (minimize RF distance) is better than optimizing the ASTRAL criterion (minimize quartet distance) for supertree estimation, and that adding bipartitions from MRL (and from ASTRID if its internode distance matrix is complete) also tends to improve accuracy.

Running time. Figure 3.2 shows running times on the biological datasets. MulRF and PluMiST took the most time, each typically requiring hours where FastRFS-basic, MRL, and ASTRAL completed in well under a minute (and sometimes in just a few seconds). MRL and FastRFS-enhanced were the next most computationally intensive, but were sometimes fast, and finally ASTRAL, ASTRID, and FastRFS-basic were the fastest, often completing in just seconds. As an example, the running times on the largest dataset on which all the methods completed (THPL, with 558 taxa) showed substantial differences between methods: PluMiST used 86400 seconds (i.e., 24 hours), MulRF used 29160 seconds (i.e., 8.1 hours), FastRFS-enhanced used 615 seconds (just over 10 minutes), MRL used 575 seconds (i.e., just under 10 minutes), and ASTRID, ASTRAL, and FastRFS-basic used under 20 seconds.

The size of X impacts the running time for FastRFS, and ranged from 1155 to 20,233 for FastRFS-basic and from 2485 to 48,313 for FastRFS-enhanced. The most computationally intensive dataset for FastRFS-enhanced is the CPL dataset, which maximizes both the number of taxa and $|X|$; however, FastRFS-enhanced completed on this dataset in 3282 seconds (i.e., under an hour). The majority of the time for FastRFS-enhanced is spent computing the MRL tree; the other parts of the analysis (i.e., computing the ASTRID matrix and potentially the ASTRID tree, computing the constraint set from ASTRAL, and running the DP algorithm) takes very little time (typically less than a minute).

ASTRID's running time was highly variable, but the running time is high only for large datasets with missing entries in the distance matrix. The reason is that when the matrix has missing entries, ASTRID must use BIONJ* (which takes $\Theta(n^3)$ time) instead of FastME (which takes $\Theta(n^2)$ time). For example, ASTRID used about 6 hours on the CPL dataset (the only biological dataset with these missing entries), but completed in just seconds on all the other datasets.

3.4 Conclusions

Supertree estimation is a basic bioinformatics challenge that is necessary for the construction of large phylogenies as well as for enabling statistical phylogeny estimation methods to be applied to large datasets. While many methods have been developed to compute supertrees, very few have been able to provide good accuracy on datasets with many hundreds or thousands of species.

The FastRFS methods presented here (i.e., the basic and enhanced versions) are fast and effective techniques to find solutions to the NP-hard Robinson-Foulds Supertree (RFS) problem. FastRFS-enhanced in particular nearly always finds better solutions than PluMiST and MulRF, the leading methods for RFS, and does so in much less time. FastRFS relies upon a dynamic programming algorithm to find an exact solution to its optimization problem within a constrained search space, a strategy introduced in [45] and that is quite different from the heuristic search techniques used by most phylogeny estimation methods. Thus, while FastRFS, PluMiST, and MulRF all seek to optimize the same criterion, FastRFS is guaranteed to find an optimal solution within its constraint space but cannot return any tree that is not within the constraint space, while PluMist and MulRF are not guaranteed to find an optimal solution within any search subspace but have access to the entire treespace. Thus, our study suggests that exactly solving an optimization problem within a constrained search space may be a better approach than being able to search a larger space, as long as the constrained space is selected carefully. However, our study also shows that expanding the constraint set beyond the input set of source trees can be highly beneficial in terms of finding good solutions to NP-hard optimization problems.

FastRFS-enhanced also tends to find more accurate tree topologies than the other supertree methods we explored. The improvement in topological accuracy suggests that the Robinson-Foulds Supertree problem is a good approach to supertree estimation. The explanation for this is likely to be the close relationship between the Robinson-Foulds Supertree problem and the Maximum Likelihood Supertree problem [19], which models source tree discord based on the topological distance to the true supertree [131]. Thus, although a Robinson-Foulds Supertree is not guaranteed to be identical to a Maximum Likelihood Supertree, good solutions to one problem are likely to be good solutions to the other [19]. Hence, FastRFS may be a good heuristic for the Maximum Likelihood Supertree problem, and this may explain its good accuracy.

There are many directions for future work. For example, since FastRFS by design can only search within the space defined by its constraint set, finding better constraint sets may provide additional improvements. Alternatively, FastRFS-enhanced may provide a good starting tree for PluMiST and MulRF, which are able to search an unconstrained search space. In addition, FastRFS-enhanced may be a good initial tree for Bayesian supertree methods [1, 2] or heuristic searches for Maximum Likelihood Supertrees [3]. Also, like most supertree methods, FastRFS currently only works with inputs where each source tree has at most one copy of each leaf; methods like MulRF are designed to handle inputs of source trees that represent gene trees, and so can have multiple copies of each species (arising from duplication-loss scenarios). We will modify FastRFS to be able to work with such source tree inputs.

Chapter 4

Improving Dynamic Programming for Phylogenomic Estimation with SIESTA

4.1 Background

Phylogeny estimation is generally approached as a statistical estimation problem, and finding the best tree for a given dataset is typically based on methods that are computationally very intensive; for example, maximum likelihood phylogeny estimation is NP-hard [114] and Bayesian MCMC methods require a long time to converge. For this reason, among others, the calculation of very large phylogenies is often enabled by divide-and-conquer methods that use "supertree methods" to combine smaller trees into larger trees. A more common use of supertree methods is to combine trees computed by independent research groups on different datasets into a single tree on a large dataset [17]. While Matrix Representation with Parsimony (MRP) [10, 110] is the most well known supertree method, other supertree methods have been shown to have better accuracy than MRP (e.g., Matrix Representation with Likelihood [105], FastRFS [144], and the recently proposed Bad Clade Deletion supertree method [37]). Supertree methods are an area of active research in the computational phylogenetics community, with new methods introduced frequently and used in a variety of contexts [2, 64, 111].

Species tree estimation, even for small numbers of species, is also difficult because of multiple processes that create differences in the evolutionary history across the genome; examples of such processes include incomplete lineage sorting (ILS), gene duplication and loss (GDL), and horizontal gene transfer (HGT) [80]. Species tree estimation is therefore performed using multiple loci from throughout the genomes of the different organisms, and is referred to as "phylogenomics". One of the standard approaches for species tree estimation is to compute gene trees (i.e., trees on different genomic regions) and then combine the trees together into a species tree under statistical models of evolution, such as the multi-species coalescent (which models ILS), that allow for gene tree heterogeneity. Examples of such "summary methods" (i.e., methods that construct species trees by combining gene trees) that are statistically consistent under the multi-species coalescent model include ASTRAL [94, 95, 156], GLASS [102], the population tree in BUCKy [66], MP-EST [75], NJst [74], and a modification of NJst called ASTRID [141].

Summary methods share algorithmic features in common with supertree methods in that both construct trees on the set of species by combining trees on subsets of the species set; the difference between the two types of methods is that in the supertree context, the assumption is that the heterogeneity observed between these "source trees" is due only to estimation error, while in the phylogenomic context the assumption is that source trees can differ from each other and from the species tree due to a combination of estimation error and true heterogeneity resulting from ILS, GDL, HGT, or some other causes. Summary methods and supertree methods are often based on attempts to solve NP-hard problems, and typically use heuristics (a combination of hill-climbing and randomization) to search for optimal trees. While these heuristics can be highly effective on small datasets, they are often very slow and there are no guarantees about the solutions they find.

An alternative approach to the use of heuristic searches is constrained exact optimization, whereby the solution space is first constrained using the input source trees, and then an exact solution to the optimization problem is found within that constrained space. This approach can lead to polynomial time methods (where the running time depends on the size of the constraint space as well as on the input) that can have outstanding accuracy. The first use of this approach was presented in [45], which provided a method to find a species tree minimizing the duplication-loss reconciliation cost given a set of estimated gene trees. Since then, many other constrained exact optimization methods have been developed in phylogenomics for different purposes, including computing trees from maximum likelihood quartet trees [20], constructing species tree from sets of gene trees under gene duplication and loss models [11] or under the multi-species coalescent model [94, 95, 139, 155], improving gene trees given a species tree [137], constructing consensus trees [20], constructing supertrees [144], and extracting a tree from a phylogenetic network [20].

Most of these approaches constrain the search space using a set of "allowed bipartitions", which we define here. Each edge e in an unrooted tree T on a set S of species defines a bipartition π_e of S (also called a "split"), obtained by deleting e but not its endpoints from T; hence, every tree T can be defined by its set of bipartitions $C(T) = \{\pi_e : e \in E(T)\}$. The constraints imposed by these algorithms are obtained by specifying a set X of allowed bipartitions so that the returned tree T must satisfy that $C(T) \subseteq X$. The set X is used to define a set of "allowed clades" (comprised of the halves of the bipartitions, plus the full set of species), and dynamic programming is then used on the set of allowed clades to find an optimal solution to the optimization problem. The set X has an impact on the empirical performance, but even simple ways of defining X can result in very good accuracy and provide guarantees of statistical consistency under statistical models of evolution [94, 144].

The constrained exact optimization approach has multiple advantages over heuristic search techniques.

From an empirical perspective, the dynamic programming approach is frequently faster, and if the constraint space is selected well it is often more accurate than alternative approaches that typically use heuristic searches for optimal solutions. From a theoretical perspective, the ability to provably find an optimal solution within the constraint space is often sufficient to prove statistical consistency under a statistical model of evolution (e.g., under the multi-species coalescent model); hence, many of the methods that use constrained exact optimization can be proven statistically consistent, even for very simple ways of defining the constraint set.

These constrained exact optimization methods typically have excellent accuracy in terms of scores for the optimization problems they address (established on both biological and simulated datasets) and topological accuracy of the trees they compute (as established using simulated datasets). A basic limitation of these methods, however, is that they return a single optimal tree, even though there can be multiple optima on some inputs. This limitation reduces the utility of the methods.

We present SIESTA (Summarizing Implicit Exact Species Trees Accurately), an algorithmic tool that can be used to enhance these dynamic programming methods for finding optimal trees. The input to SIESTA is the set $\mathcal{T}$ of source trees, the constraint set X of allowed bipartitions, and a scoring function w that assigns scores to tripartitions of the taxon set (and which is derived from the optimization function F that assigns scores to trees and the set $\mathcal{T}$, as we show later); SIESTA returns a data structure $\mathcal{I}$ that represents the set $\mathcal{T}^*$ of trees that optimize the function F subject to the constraint that every bipartition in every tree in $\mathcal{T}^*$ is in X. This data structure $\mathcal{I}$ enables the user to explore the set of optimal trees in various ways. In this study, we use SIESTA to compute consensus trees, to enumerate the set of optimal trees, to count the number of optimal trees, and to report the frequency of each bipartition in the set of optimal trees.

We explore the impact of using SIESTA with two methods that use dynamic programming for constrained exact optimization: the supertree method FastRFS [144] and the ILS-aware species tree estimation method ASTRAL [94]. We show that using SIESTA to compute a strict consensus tree provides improvements in accuracy (in terms of the topology of the estimated tree) compared to a single optimal tree for both ASTRAL and FastRFS when the number of optimal trees is large enough, and is otherwise neutral. Furthermore, using SIESTA with a modification to FastRFS produces more accurate rooted supertrees than Bad Clade Deletion (BCD), the previous best method for rooted supertree construction [37].

Using SIESTA with ASTRAL, a species tree estimation method that addresses incongruence due to ILS, provides additional benefits. For each optimal tree it returns, ASTRAL provides branch support values based on local posterior probabilities, but these values do not take the other optimal trees into account. We show how to correct these support values to take the full set of optimal ASTRAL trees into account, and enable the calculation of a maximum clade credibility (MCC) tree based on these corrected values. Hence, SIESTA provides a valuable tool for both species tree and supertree estimation, providing distinct advantages over the simplistic use of leading methods for these problems. SIESTA, combined with ASTRAL and FastRFS is available at https://github.com/pranjalv123/SIESTA and the datasets analyzed in this paper are available at [140].

4.2 Methods

4.2.1 The SIESTA Algorithm

SIESTA is designed to work with tree estimation methods that seek optimal solutions within a constrained search space using dynamic programming. Recall that in the constrained optimization approach, the input is a set of source trees (estimated gene trees in the case of ASTRAL, generic source trees in the case of FastRFS) as well as a set X of allowed bipartitions of the set S of species. Given this set X of allowed bipartitions, we define a set $\mathcal{C}$ of "allowed clades" by taking the two halves of each bipartition, and we also include the set S; thus, $\mathcal{C} = \{A : [A|S \setminus A] \in X\} \cup \{S\}$.

We also form a set TRIPS of "allowed tripartitions", as follows. TRIPS contains all ordered 3-tuples (A, B, C) of allowed clades that are pairwise disjoint, that union to S, and where $A \cup B$ is also an allowed clade. We require that A and B be non-empty, but we allow C to be empty.

The purpose of creating this set is that it allows us to perform the dynamic programming algorithm to find optimal solutions for some optimization problems. To see this, consider an unrooted binary tree T that is a feasible solution to the constrained optimization problem under consideration. Now root the tree T arbitrarily and pick some internal node v defining clade c. Since T is a feasible solution to the optimization problem, all the clades in $T^{(r)}$ (the rooted version of T) are allowed clades, and every node v defining clade c that is not a leaf has two major subclades A and B defined by its two children. The 3-tuple (A, B, C) where $C = S \setminus (A \cup B)$ is the tripartition associated to node v (equivalently, associated to clade c). If v is the root of T, then C will be empty. The set of "allowed tripartitions" is defined to ensure that it includes all 3-tuples that could be formed in this way. Finally, by construction, we consider (A, B, C) and (B, A, C) to be equivalent tripartitions. Similarly, given a rooted binary tree $T^{(r)}$ on leafset S, each non-leaf node v in $T^{(r)}$ defines a tripartition (A, B, C) where A and B are the clades (i.e., leafsets) below the two children of v, and $C = S \setminus (A \cup B)$. We refer to the set of tripartitions of a rooted binary tree $T^{(r)}$ by trips($T^{(r)}$).

The objective of the constrained optimization problems is to find an unrooted tree T^* on leafset S that optimizes a function $F(\cdot)$ defined on unrooted trees, subject to T^* drawing its bipartitions from X. Hence, if we root T^*, we obtain a rooted tree $T^{*(r)}$ in which the non-leaf nodes define allowed tripartitions. ASTRAL and FastRFS are each algorithms that find optimal binary trees for some optimization problem, subject to the constraint that the tree draw its bipartitions from a set X of allowed bipartitions. These algorithms reframe the problem by seeking a rooted tree that draw its clades (i.e., subsets of leaves defined by internal nodes) from the set $\mathcal{C}$ of allowed clades, and use the dynamic algorithm design that we will now describe.

For both ASTRAL and FastRFS, it is possible to define a function w on allowed tripartitions such that for any unrooted binary tree T on leafset S, letting T^r denote a rooted version of T (obtained by rooting T on any edge),

$$F(T) = \sum_{t \in \text{trips}(T^r)} w(t) \tag{4.1}$$

where $F(T)$ is the optimization score for tree T.

The existence of a function w that is defined on tripartitions and that satisfies Equation 4.1 is the key to these dynamic programming algorithms. Given function w that is defined on tripartitions, we define a recursive function f that is defined on clades that we can then use to find optimal solutions. We show how to define f for a maximization problem; defining it for a minimization problem is equivalently easy.

The calculation of $f(c)$ for a given allowed clade c given w and X uses the following recursion (phrased here in terms of maximization):

$$f(c) = \begin{cases} \max\{f(a) + f(b) + w(a,b,x) | (a,b,x) \in \text{TRIPS}, a \cup b = c\}, & |c| > 1 \\ 0, & |c| = 1 \end{cases}$$

By Equation 4.1, $f(S) = F(T^*)$, where T^* is the optimal solution to the constrained optimization problem.

Hence, we can solve the optimization problem using dynamic programming. We compute all the $f(c)$ from the smallest clades to the largest clade S. To construct the optimal solution T^*, when we compute $f(c)$ for a clade c, we record how we obtained this best score (i.e., we record the unordered pair (a, b) of clades whose union is c achieving this optimal score), and we use backtracking to construct the rooted version of T^*. Then we unroot the rooted tree.

The SIESTA data structure

SIESTA modifies these algorithms so they output a data structure that implicitly represents the set of all the optimal trees.

Specifically, when SIESTA computes $f(c)$, instead of recording a single split of the clade c into two subclades that achieves the optimal score for the clade c, SIESTA records all such splits of c. We describe the high-level idea of SIESTA by describing how a single optimal tree (all of whose clades are drawn from

$\mathcal{C}$) can be represented with pointers, and then show how to extend that to represent all optimal trees.

Let T be a rooted binary tree, all of whose clades are drawn from $\mathcal{C}$. T can be stored as a collection of nodes, where each node contains either two pointers (one to each of its two children, if it is an internal node) or a taxon label (if it is a leaf node). Equivalently, this representation of T can be seen as having pointers from each clade c (with at least two species) to a pair of disjoint clades c_1 and c_2, whose union is c.

We modify this representation to compactly represent a set of rooted binary trees, as follows. Recall that during the dynamic programming algorithm, all optimal ways of splitting a clade c into two clades c' and $c'' = c \setminus c'$ are determined. Each of these ways of splitting c into two subclades is stored in a set $\mathcal{I}(c)$, by having each such split represented by a pair of pointers. In other words, instead of having each clade have a pair of pointers to two subclades, each clade has a set $\mathcal{I}[c]$ of pairs of pointers to a potentially large number of subclades. Thus, the SIESTA data structure is the array $\mathcal{I}$ indexed by the clades in $\mathcal{C}$, and each element of the array is a set. Note also that $|\mathcal{I}(c)| \leq |X|$, so that the SIESTA data structure uses $O(|X|^2)$ space.

The SIESTA data structure also naturally defines a directed acyclic graph whose nodes are labelled by allowed clades c (i.e., elements of X), and there is an edge from c to c' if the set $\mathcal{I}(c)$ contains a pair of pointers, with one pointer pointing to c'. We will say that c' is a child of c when there is an edge from c to c'. Given such a representation, it is easy to generate any single optimal tree by following a tree from the root of the SIESTA digraph (i.e., starting with the entry $\mathcal{I}[S]$) down to the leaves, and at each clade x with at least two elements, picking a pair of its children whose clades union to x.

The asymptotic running time of this phase is equal to the asymptotic running time of the original DP algorithm, which is $O(|X|^2\alpha)$, where α is the time required to calculate w for a single tripartition [95]. Storing the entire data structure requires $O(|X|^2)$ space in the extreme case where every tree has the same score, but in many real-world cases will require less.

4.2.1.1 Using SIESTA

We show how we can use SIESTA in various ways, including counting the number of optimal trees, generating greedy, strict, and majority consensus trees, and computing the maximum clade credibility tree.

Counting the number of optimal trees. We traverse the collection of allowed clades from smallest to largest, calculating for each allowed clade c the number optsubtrees(c) of optimal rooted binary trees that contain exactly the taxa in c. Obviously, optsubtrees(c) = 1 for all clades of size 1. It is also straightforward to check that the number of optimal rooted binary subtrees on larger clades can be computed by examining all the optimal splits of the clade into two parts. Hence,

$$\text{optsubtrees}(c) = \begin{cases} \sum_{(x,y)\in\mathcal{I}[c]} \text{optsubtrees}(x) \cdot \text{optsubtrees}(y), & |c| > 1 \\ 1, & |c| = 1 \end{cases}$$

The number of optimal rooted binary trees is optsubtrees(S), where S is the entire set of species. For the algorithms we consider (ASTRAL and FastRFS), all rootings of a particular unrooted tree have the same criterion score, and so this quantity should be divided by $2n-3$, where $n = |S|$ is the number of species, to get the number of optimal unrooted trees.

Calculating consensus trees. A particular bipartition $[c|S \setminus c]$ is present in fraction A_c of the optimal trees, where

$$A_c = \frac{\text{optsubtrees}(c) * \text{optsubtrees}(S \setminus c)}{\text{optsubtrees}(S)} \tag{4.2}$$

For $\alpha \geq 0.5$, the α-consensus tree is the unique tree that contains exactly those bipartitions that occur in more than fraction α of the optimal trees. For smaller values of α, we can still construct a consensus tree, but the set of bipartitions that appear with frequency greater than α may not form a tree. To construct the α-consensus tree, we sort the bipartitions in descending order by A_c, restricted only to those bipartitions $[c, S \setminus c]$ with $A_c > \alpha$, and construct a greedy consensus tree using this ordering. To calculate a greedy consensus tree, we sort all the bipartitions in descending order of A_c and greedily build a tree from them. The majority consensus tree has $\alpha = 0.5$, and so is an example of an α-consensus tree. The strict consensus tree can also be computed easily, and contains only the bipartitions that It is easy to see that each of these consensus trees can be computed in $O(|X| \log |X|)$ time.

Correct local branch support in an ASTRAL tree. Recall that ASTRAL-II uses a quartet-based local posterior probability (PP) measure [121] to assign support values to edges. However, when there is more than one optimal tree, the branch support in any individual tree is unreliable, since it does not take the other optimal trees into account. However, SIESTA can modify the branch support values by taking the other optimal trees into account. Specifically, for a given bipartition in a tree T, we compute its average support across the set of optimal trees (where an optimal tree without the bipartition contributes a support of zero); this is the corrected support for the bipartition.

The ASTRAL Maximum Clade Credibility tree. A natural optimization problem would be to return the tree whose total corrected branch support (as described above), summed over all the edges of the tree, is maximized. Such a tree is called the Maximum Clade Credibility (MCC) tree, but finding such a tree is an NP-hard problem. We developed a greedy heuristic for the MCC tree, as follows. We use SIESTA to compute every optimal ASTRAL tree, and calculate the corrected local branch support values (as described above). We then compute a greedy consensus of the resulting bipartitions, ranked by these corrected support values. We refer to this as the ASTRAL MCC tree.

4.2.2 Evaluation Protocol

We tested SIESTA in two contexts: in conjunction with FastRFS (a supertree method) and in conjunction with ASTRAL (an ILS-aware species tree estimation method). We use both biological and simulated datasets for these experiments, and on each dataset we examined, we used SIESTA to compute the set of optimal solutions, and to compute consensus trees for these sets of optimal trees. Overall, we examined 1020 simulated and 16 biological datasets (5 supertree and 11 phylogenomic).

Gene tree estimation. The simulated supertree datasets (both rooted and unrooted) and all the biological datasets we analyzed came with pre-calculated source trees; for the other datasets (i.e., for the simulated phylogenomic datasets) we used RAxML v8.2.4 [129] to estimate gene trees (using options `-m GTRGAMMA -p 12345`).

Supertree methods. We evaluated the impact of SIESTA on the FastRFS v2.0 supertree method, using several variants of FastRFS that vary in how the constraint set of allowed bipartitions is defined:

- FastRFS$_{basic}$, which only uses ASTRAL-II to compute the constraint set,
- FastRFS$_{enh}$ (i.e., the enhanced version), which adds the bipartitions from the Matrix Representation with Likelihood (MRL) supertree to its constraint set and also from the ASTRID tree (but only when the internode distance matrix that ASTRID computes is complete), and
- FastRFS$_{BCD}$, which adds the bipartitions from the BCD supertree, but can only be used with rooted supertree datasets.

Hence, FastRFS uses other supertree methods (i.e., ASTRAL, MRL, ASTRID, and BCD) to compute the constraint set. We ran ASTRID v1.1 and BCD v1.0.1 in default mode. For ASTRAL-II, we ran a custom variant (available at the github site) where we use ASTRAL v4.7.8 to compute the constraint set of allowed bipartitions, and then our own dynamic programming implementation to find optimal solutions to the quartet support optimization problem. This custom version (which we call SIESTA-ASTRAL) produces exactly the same output species tree(s) as ASTRAL v.4.7.8, and allows us to make a comparison between SIESTA used with ASTRAL v4.7.8 to compute consensus trees and a single ASTRAL 4.7.8. tree. For MRL, we used RAxML v8.2.4 [129], with options `-m BINGAMMA -p 12345`.

The supertree FastRFS$_{enh}$ has already been shown to produce more accurate supertrees than ASTRID, ASTRAL, and MRL, on simulated datasets [144]. However, a new supertree method, BCD, has been developed for use with rooted source trees, and has been reported to be more accurate than FastRFS; hence, we explore these FastRFS variants on supertree datasets with rooted source trees, and we compare these variants to BCD. We then explore the impact of SIESTA on the best variant and determine how it compares to BCD.

ILS-aware species tree methods. We evaluated the impact of SIESTA on ASTRAL v4.7.8 on the phylogenomic datasets. We also used ASTRID, v1.1 (another ILS-aware method), but only in the context of providing bipartitions for FastRFS. For the biological datasets, we explored the use of the MCC (Maximum Clade Credibility) tree computed using SIESTA.

Consensus methods. For each dataset, we use SIESTA to compute the set of optimal trees and then also to compute three consensus trees: the strict consensus, the majority consensus, and the greedy consensus. The strict consensus tree is the unique tree whose bipartition set is exactly those bipartitions that appear in every optimal tree, and so will not be fully resolved whenever the number of optimal trees is two or larger. The majority consensus tree is the unique tree whose bipartition set is exactly those bipartitions that appear in a strict majority of the set of optimal trees; unlike the strict consensus, it may be fully resolved even when there are two or more optimal trees. Finally, the greedy consensus tree is obtained by ordering the bipartitions according to their frequency in the set of optimal trees, and then adding them, one by one, in order of their frequency (from most frequent to least frequent) to a growing tree. By design, the greedy consensus may not be unique, but will always refine (or equal) the majority consensus; similarly, the majority consensus will always refine (or equal) the strict consensus.

4.2.2.1 Datasets

Simulated supertree datasets. We use two collections of simulated supertree datasets (one with unrooted source trees and one with rooted source trees), each based on the SMIDgen [133] simulation protocol. The unrooted source trees were originally generated for [133], and have been used to explore the accuracy of several supertrees methods [105, 144]; the rooted source tree datasets were generated for [37], and enable a comparison with the BCD supertree method [37], which requires rooted source trees.

We explore the results on the datasets with 100, 500, and 1000 taxa. Each replicate contains one "scaffold" tree and several clade-based trees. The scaffold tree is based on a random sample of the species, and contains 20%, 50%, 75%, or 100% of the taxa sampled uniformly at random from the leaves of the tree. The clade-based trees are based on a clade and then a birth-death process within the clade (and hence may miss some taxa). The original 100-taxon, 500-taxon, and 1000-taxon datasets had 6, 16, and 26 source trees respectively; the number of source trees was reduced to 6, 11, and 16 for the 500-taxon datasets, and 6, 11, 16, 21, and 26 for the 1000-taxon datasets. Sequences evolved down each scaffold and clade-based source tree under a GTR+Gamma model (selected from a set of empirically estimated parameters) with branch lengths that are deviated from the strict molecular clock. Maximum likelihood trees were estimated on each sequence alignment using RAxML under the GTRGAMMA model (with numeric parameters estimated by RAxML from the data), and used as source trees for the experiment. 25 replicates were analyzed for the 100- and 500-taxon model conditions, and 10 replicates were analyzed for each scaffold factor of the 1000-taxon model condition.

Simulated phylogenomic datasets. We obtained multi-locus simulated datasets from [94], and then modified them for this study. These datasets were generated by evolving gene trees within species trees (with speciation close to the leaves of the model tree) under the multi-species coalescent (MSC) model using SimPhy [83], and then evolving sequences down each gene tree under the GTR+Gamma model, with branch lengths deviated from the strict molecular clock, using Indelible [38]. Three levels of ILS were generated by modifying the species tree height.

These datasets were then modified for the purposes of this study. These datasets originally had 200 taxa each, but were randomly reduced to 50 taxa each to reduce the running time. The original datasets had variable length loci between 300 and 1500bp, and were truncated for this experiment to 150bp to produce datasets with properties that are consistent with empirical phylogenomic datasets (which frequently have very low phylogenetic signal). Each replicate was evaluated with 5, 10, and 25 loci. We evaluated model conditions where each gene contained all 50 taxa, as well as model conditions where each gene contained 10, 20, or 30 taxa chosen at random from the taxon set. These datasets with 50 taxa had ILS levels that ranged from moderate to very high; we characterize the ILS using the average normalized bipartition distance (AD) between true gene trees and true species trees. The moderate ILS condition has AD=12%, the high ILS condition has AD=31%, and the very high ILS condition has AD=68%. We also generated incomplete gene trees by randomly deleting a specific number of taxa from each gene (so that all genes are incomplete but have the same number of leaves) and then re-estimated gene trees; this allows us to evaluate species tree estimation when not all genes have all the species (i.e., in the presence of "missing data") [100].

We estimated gene trees using RAxML [129] under the GTRGAMMA model (with numeric parameters estimated by RAxML), and we analyzed 25 replicates for each model condition (defined by the ILS level, number of loci, and amount of missing data).

Biological supertree datasets. We analyzed five (all unrooted) supertree datasets from [133]: Marsupials [23], Placental Mammals [15], Seabirds [56], Temperate herbaceous papilionoid legumes (THPL) [152], and Comprehensive papilionoid legumes (CPL) [87] datasets. See Table 4.1 for detailed information about these datasets.

Biological phylogenomic datasets. We analyzed 11 phylogenomic datasets, described in Table 4.2. Each of these datasets has multiple genes, and each gene has one unrooted binary maximum likelihood gene tree.

4.2.2.2 Performance criteria.

For the simulated datasets, we compare the topological accuracy of the trees we compute by comparing them to the model species tree or supertree. We use DendroPy v4.0.3 [132] to compute both the false negative (FN) rate and the false positive (FP) rate with respect to the model tree, where the FN rate is the number of bipartitions in the model tree that are missing from the estimated tree and the FP rate is the number of bipartitions in the estimated tree that are not in the model tree, each divided by $n - 3$ (the number of internal edges in an unrooted tree) where n is the total number of leaves in the model tree. For each basic tree estimation method (i.e., ASTRAL and FastRFS), we also report Delta-Error, which is the difference between the average error rate (i.e., the average of the FN and FP error rates) computed for the tree estimation method and the average error rate of the strict consensus of the optimal trees found by that method. Hence, when Delta-Error is negative, the strict consensus has overall lower error than a single optimal tree. We also report the $F1$ score, which is the harmonic mean of the precision and recall of the estimated trees. For the biological datasets, since topological accuracy cannot be assessed exactly, we describe differences between the consensus trees we compute using SIESTA and trees computed using other techniques. We also report the number of optimal trees for the optimization problems on all the datasets we examine, and the running time used on the biological datasets.

4.3 Results and Discussion

4.3.1 Overview

Experiment 1 explores the use of SIESTA to compute the number of optimal trees found by FastRFS and ASTRAL, as this indicates the potential for SIESTA to improve accuracy by computing consensus trees. Experiment 2 explores how the choice of consensus tree (strict, majority, or greedy) impacts the average topological accuracy of the resulting tree. The next experiments compare the strict consensus tree to a single optimal tree, with Experiment 3 examining FastRFS variants on simulated supertree datasets and Experiment 4 examining ASTRAL on simulated phylogenomic datasets. Experiment 5 examines the use of SIESTA to calculate branch support with ASTRAL and FastRFS on biological datasets, and Experiment 6 evaluates running time issues.

4.3.2 Experiment 1: Computing the number of optimal trees

We used SIESTA to compute the number of optimal trees found by FastRFS and ASTRAL on both the biological and simulated datasets. We explore the differences between FastRFS variants (which depend on how the constraint set is defined) and also between FastRFS and ASTRAL.

FastRFS variants. As shown in Table 4.1, FastRFS$_{enh}$ tends to produce large numbers of optimal trees on the biological supertree datasets, and this number tends to increase with the number of taxa and decreases with the number of source trees. On the simulated supertree datasets, both FastRFS$_{enh}$ and FastRFS$_{basic}$ typically have a large number of optimal trees (Additional file 1, Tables S1 and S2), but FastRFS$_{enh}$ generally had a much larger number of optimal trees than FastRFS$_{basic}$. In addition, the number of optimal trees for both variants grows with the number of taxa: FastRFS$_{enh}$ typically has tens or hundreds of optimal solutions on datasets with 100 taxa, but there are up to 10^{18} optimal FastRFS$_{enh}$ trees on datasets with 1000 taxa. The density of the scaffold factor also impacts the number of optimal trees, with fewer optimal trees with the 100%-scaffold factor than with sparser scaffold factors.

ASTRAL. ASTRAL showed distinctly different trends. For example, ASTRAL typically only produced a single optimal tree on the biological phylogenomic datasets, as shown in Table 4.2. We also examined the number of optimal ASTRAL trees on simulated phylogenomic datasets. As shown in Additional file 1, Table S3, when all the gene trees are complete, nearly all the analyses produced only one optimal ASTRAL tree, and when more than one tree was produced it was typically a very small number (often just two). However, there are many optimal ASTRAL trees on the phylogenomic datasets with incomplete gene trees (see Additional file 1, Table S4). Thus, although ASTRAL usually only finds a single optimal tree, it can (in some cases) return a larger number.

Comparison of ASTRAL and FastRFS variants on the same datasets. We then compared the number of optimal trees found by ASTRAL, FastRFS$_{basic}$, and FastRFS$_{enh}$ on the biological supertree datasets. FastRFS$_{enh}$ found the largest number, followed by FastRFS$_{basic}$, and then by ASTRAL (Table 4.3). The comparison between FastRFS$_{basic}$ and FastRFS$_{enh}$ shows that increasing the size of the constraint space for FastRFS results in an increase in the number of optimal trees, which is as expected.

The comparison between ASTRAL and FastRFS$_{basic}$, which have the same constraint set, is more interesting, and suggests that the optimization problem solved by ASTRAL tends to have a smaller set of optimal trees than the optimization problem solved by FastRFS. The reason that FastRFS tends to have more optimal solutions than ASTRAL may be that the number of possible FastRFS scores is substantially smaller than the number of possible ASTRAL scores. Specifically, if n is the number of species and k is the number of source trees, the FastRFS scores are all integers in the range $[0, (n-3)k]$, while the possible ASTRAL scores are integers in the range $[0, k\binom{n}{4}]$. Therefore, the frequency of multiple trees with the same optimal score is higher for FastRFS than for ASTRAL. However, ASTRAL has by far a much smaller number of optimal trees, and typically has only one optimal tree under conditions where even FastRFS$_{basic}$ has at least 10^6 optimal trees.

Overall, therefore, FastRFS$_{enh}$ typically has many optimal trees on supertree datasets, while ASTRAL typically (but not always) has only one optimal tree when given complete gene trees but can have many optimal trees when given highly incomplete gene trees. This means that if we use SIESTA to compute a consensus tree of the optimal trees, this has a greater probability of impacting FastRFS$_{enh}$ than ASTRAL, but can also impact ASTRAL when the input dataset has genes that are missing many taxa.

4.3.3 Experiment 2: Comparing different consensus trees computed using SIESTA

We explored the impact of using different consensus methods (i.e., the strict consensus, majority consensus, and greedy consensus) in conjunction with FastRFS$_{enh}$ and FastRFS$_{BCD}$. We report the difference in average topological error (i.e., the average of the FN and FP error rates) of these consensus trees compared to a single best tree.

For the unrooted supertree datasets, as seen in Additional file 1, Figure S1, for all numbers of taxa and scaffold factors, the three consensus trees of the best FastRFS$_{enh}$ supertrees are nearly identical in accuracy, and typically are more accurate than a single best FastRFS$_{enh}$ tree. However, there are some cases where the strict consensus has a very slight advantage over the other consensus methods. Additional file 1, Figure S2 shows FN and FP rates separately for the strict consensus of the optimal FastRFS$_{enh}$ trees on the unrooted supertree datasets, and how they are impacted by the number of optimal trees. As expected, the FP rates decrease and the FN rates increase as the number of optimal trees increases; furthermore, as the number of optimal trees increases, the decrease in FP rate is substantially larger than the increase in FN rate. As a result, the average of the FN and FP rates decreases with the number of optimal trees.

We then explored the impact of choice of consensus tree on the simulated rooted supertree datasets (where we used FastRFS$_{BCD}$); see Additional file 1, Figure S3. On these data, the strict consensus tree had generally the lowest average topological error rate, followed by the majority consensus, and then by the greedy consensus, but all three consensus trees were typically more accurate than a single best FastRFS$_{BCD}$ tree.

4.3.4 Experiment 3: FastRFS-SIESTA vs. FastRFS on simulated supertree datasets

We compare the strict consensus of the optimal FastRFS supertrees (referred to as FastRFS-SIESTA) to a single FastRFS supertree on the simulated supertree datasets. For the unrooted supertree datasets, we use FastRFS$_{enh}$, which was shown to provide better topological accuracy than other supertree methods in [144].

Results on the unrooted supertree datasets (Fig. 4.1) show that FastRFS+SIESTA is at least as accurate as FastRFS for all scaffold factors and all numbers of taxa. The difference between the two methods is often small, but there are larger improvements when the scaffold factor is the smallest (which is also when the number of optimal trees is largest).

For rooted supertree datasets, we explore another supertree method called the Bad Clade Deletion (BCD) supertree method, which can only be used with rooted source trees. As shown in [37], BCD produced more accurate species trees (with respect to the F1 metric) than FastRFS$_{basic}$ and several other supertree methods. We confirm that BCD outperforms FastRFS$_{basic}$ with respect to the F1 metric (Additional file 1, Fig. S4), and also note that BCD outperforms FastRFS$_{basic}$ with respect to the RF error rate (Additional file 1, Fig. S5). However, it is not known whether BCD is more accurate than FastRFS$_{enh}$ or FastRFS$_{BCD}$, nor whether using SIESTA enables some FastRFS variant to outperform BCD. We compared these three methods with respect to RF errors (Additional file 1, Fig. S6) and F1 scores (Additional file 1, Fig. S7). The two FastRFS variants are very close in accuracy with respect to both criteria, with a slight advantage to FastRFS$_{BCD}$. Interestingly, the comparison to BCD shows that the FastRFS variants are less accurate on the sparse scaffolds than BCD, but slightly more accurate on the 100%-scaffold. Overall, therefore, FastRFS$_{BCD}$ has a slight advantage over the other FastRFS variants on these rooted supertree datasets, and is competitive with BCD (worse under some conditions and better under others).

We then examined whether computing the strict consensus improves FastRFS$_{BCD}$ enough to enable it to outperform BCD. We first observed that the strict consensus of the FastRFS$_{BCD}$ supertrees was more accurate than a single FastRFS$_{BCD}$ supertree (Fig. 4.2). Furthermore, using SIESTA to compute the strict consensus of the optimal trees found by FastRFS$_{BCD}$ produces supertrees that are generally (but not always) more accurate than BCD (Fig. 4.3 shows average tree error and Additional file 1, Fig. S8 shows the F1 scores). The differences are smallest on the 100-taxon datasets, but the strict consensus of the FastRFS$_{BCD}$ trees is generally more accurate than BCD on the larger datasets, especially for the denser scaffolds. Thus, the use of SIESTA enables FastRFS$_{BCD}$ to outperform BCD.

4.3.5 Experiment 4: ASTRAL+SIESTA vs. ASTRAL on simulated phylogenomic data

As noted earlier, ASTRAL often returns only one optimal tree, so that the strict consensus of the optimal ASTRAL trees cannot differ from the single best tree. In this experiment, we restrict the attention to the datasets on which ASTRAL found more than one tree. In general, this occurred for the phylogenomic datasets with substantial levels of missing data (i.e., when we deleted species randomly from genes). For these cases, we see that the average topological error rates for the strict consensus of the ASTRAL trees are lower than the error rate for a single ASTRAL tree (Fig. 4.4) under three different ILS levels, when there is missing data. However, the degree to which the strict consensus of the ASTRAL trees improves over a single ASTRAL depends upon the amount of missing data.

A more nuanced analysis is shown in Figure 4.5, where we explore how the number of optimal trees impacts the FN and FP rates for the strict consensus. Note that the FN rate of the strict consensus is very similar to the FN rate of a single optimal ASTRAL tree, but the strict consensus has a much lower FP rate; hence the strict consensus has a reduced average error rate compared to a single best tree. Although the FN rates are slightly higher under lower ILS conditions, the FP rates drop more than the FN rates increase, so that the same overall trends are similar (Additional file 1, Fig. S9).

4.3.6 Experiment 5: Results on biological datasets

For the biological datasets, we do not know the true species tree (which is the unstated objective of the supertree analysis), and so we cannot evaluate accuracy. However, we show how to use SIESTA to provide meaningful branch support in estimated species trees.

Biological supertree datasets. We use SIESTA to compute the greedy consensus tree of the FastRFS$_{enh}$ supertrees on the unrooted supertree datasets, and then annotated each edge in the greedy consensus supertree with the fraction of the optimal trees on the dataset. Figure 4.6 shows that most of the edges in the greedy consensus of the optimal FastRFS$_{enh}$ supertrees for each of these datasets have 100% support, indicating that these edges are consistent across all optimal trees. It also shows that some edges are only found in about half (sometimes even less) of the optimal trees, and so should not be considered as reliable. However, this depends on the dataset: nearly all the edges in the greedy consensus of the optimal FastRFS$_{enh}$ supertrees for the placental mammals dataset have 100% support, while the THPL and CPL datasets have a substantial fraction of edges that appear in at most 60% of the optimal FastRFS$_{enh}$ supertrees.

Hymenoptera phylogenomic dataset. The Hymenoptera dataset is a phylogenomic dataset with 21 taxa and 24 genes. There are four optimal ASTRAL trees on this dataset (shown in Fig. 4.7). The differences between these four trees are restricted to two clades with three species each: (1) Solenopsi, Apis, and Vesputal_C, and (2) Acyrthosi, Myzus, and Acyrthosp. The strict and majority consensus trees (Fig. 4.8) on these four ASTRAL trees are identical, and present these two groups as completely unresolved. The MCC tree (Fig. 4.8) on this set of four ASTRAL trees matches one of the four trees with respect to topology, but has different branch support on the edges, so that the branch support for the two clades in question are halved in comparison to the four ASTRAL trees; thus, the MCC tree appropriately identifies these clades as having very low support.

Sigmontidine rodent phylogenomic dataset. The Sigmontidine rodent dataset is a phylogenomics dataset with 285 taxa and 11 genes, and there are 72 optimal ASTRAL trees on this dataset. The species tree computed on this dataset in [81] was a concatenated Bayesian tree using MrBayes [118], with branch support based on posterior probabilities. The Sigmontidine rodent dataset had 72 optimal ASTRAL trees. We computed the ASTRAL MCC tree, and then collapsed all branches with support less than 75%; this produced a tree with only 74 internal edges. This dataset has 285 taxa, meaning that a fully resolved tree would have 282 internal branches. By comparison, the MrBayes tree has 223 internal branches after collapsing branches with less than 75% support.

Comparing the MrBayes tree with the ASTRAL MCC tree, we find that 64 bipartitions are present and highly supported in both trees. After collapsing the edges with lower support, we are left with only the high support edges. Six highly supported bipartitions are present in the ASTRAL MCC tree and compatible with the collapsed MrBayes tree, and three bipartitions are present in the ASTRAL MCC tree and incompatible with the collapsed MrBayes tree. 153 highly supported bipartitions are present in the MrBayes tree and compatible with (but not present in) the collapsed ASTRAL MCC tree, and 5 highly supported bipartitions in the MrBayes tree are incompatible with the collapsed ASTRAL MCC tree. The highly supported conflicts between the trees occur in three locations:

1. The MrBayes tree has *Akodon Mimus* as the root of the *Akodon* genus, while the ASTRAL MCC tree has it internal to *Akodon* (the root of *Akodon* is not resolved with greater than 75% support).

2. The MrBayes tree and the ASTRAL MCC tree swap the locations of the *Holochilus* and *Sooretamys* clades, with ASTRAL putting *Holochilus* as the basal clade and MrBayes putting *Sooretamys* as the basal clade.

3. The ASTRAL MCC tree and the MrBayes tree disagree about some resolutions within the *Oligoryzomys* clade.

These placements are in general not well established in the literature [7, 43, 79], and so it is not clear which of the two trees is more likely to be correct for these questions.

The difference between a single ASTRAL tree and the ASTRAL MCC tree is therefore quite significant for some datasets. To understand these differences, recall that the support values are obtained using posterior probabilities based on quartet trees around an edge in a single optimal tree. However, a simple example can explain why this can be misleading. Suppose T_1 and T_2 are the only trees that are optimal for ASTRAL, and that T_1 has a split π that T_2 does not have. Then under the assumption that T_1 and T_2 are both equally likely to be the true species tree, the *maximum* probability that π can be a true split is 0.5 – since it is in only one optimal tree. It is easy to see that any support value greater than 0.5 produced when T_1 is examined is inflated, and that a correction must be made that takes into consideration that T_2 is also an optimal tree. SIESTA's way of calculating support explicitly enables this correction, since it explicitly considers the support of each bipartition obtained from the entire set of optimal trees.

4.3.7 Experiment 6: Running Time

We explore the computational impact of using SIESTA to compute the strict consensus of the optimal trees found using two variants of FastRFS on the rooted supertree datasets with 1000 species. We compare the cost of using FastRFS$_{basic}$ to find a single tree to the total running time needed to compute the strict consensus of the FastRFS$_{basic}$ supertrees (Table 4.4). All methods complete in under a minute (actually under 40 seconds), and that the difference in terms of time needed to compute a single FastRFS$_{basic}$ tree and the strict consensus of all the optimal FastRFS$_{basic}$ trees is at most 0.3 seconds. We also compare the time needed to run BCD, FastRFS$_{BCD}$, and the total time needed to compute the strict consensus of the FastRFS$_{BCD}$ supertrees (Table 4.5). Note that BCD is substantially faster than FastRFS$_{BCD}$, but that all methods complete in less than a minute. Note also that the difference in terms of time needed to compute a single FastRFS$_{BCD}$ tree and the strict consensus of all the optimal FastRFS$_{BCD}$ trees is at most 0.5 seconds

Thus, the additional time needed to compute the strict consensus of the set of optimal trees is less than half a second. This is particularly noteworthy, given the number of optimal trees that are found by FastRFS$_{basic}$ on these 1000-taxon supertree datasets. Overall, these data show that the cost of using SIESTA is small, and represents a small percentage of the total time needed to find a single tree.

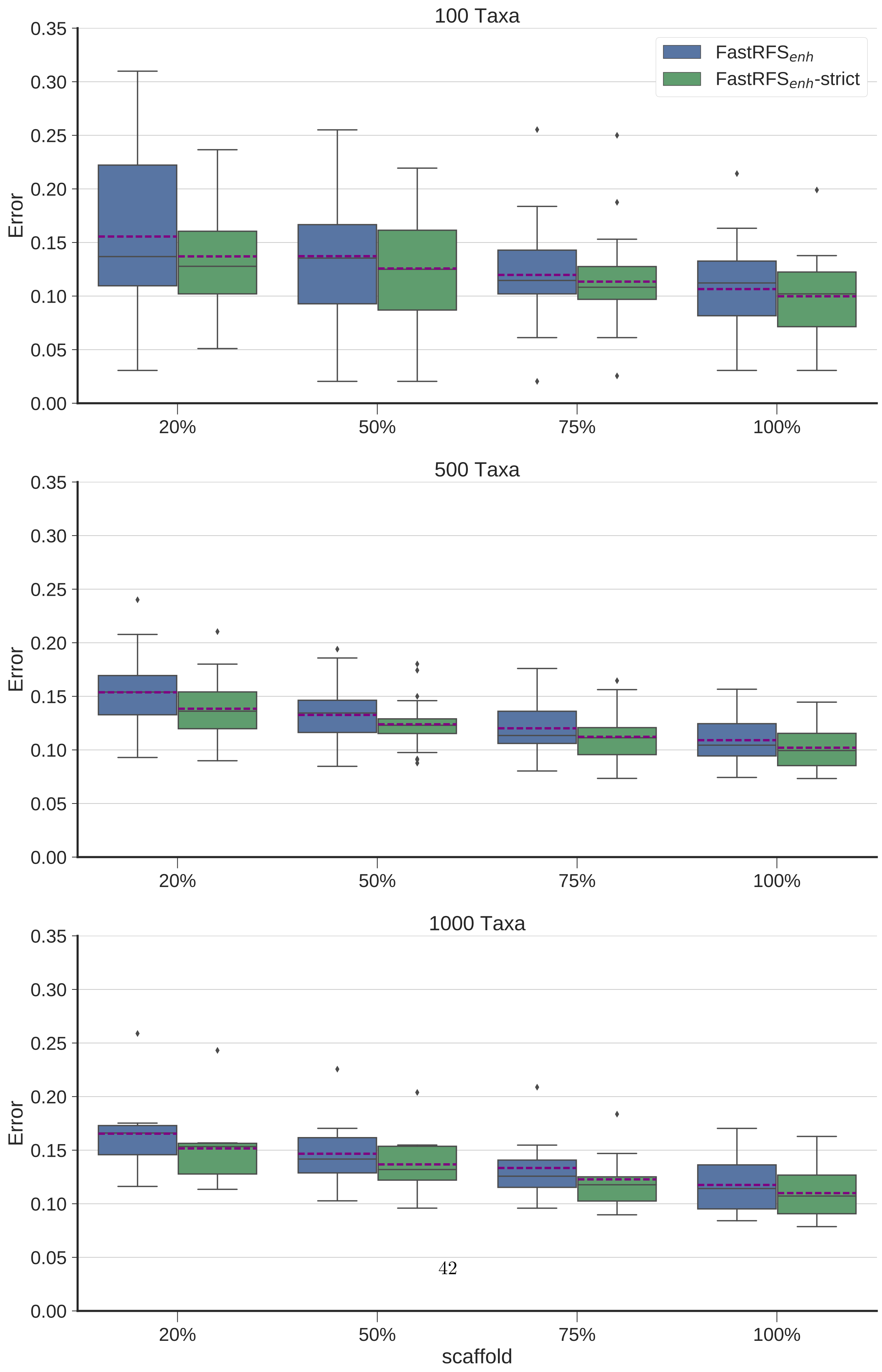
4.4 Conclusions

SIESTA is a simple technique for computing a data structure that implicitly represents a set of optimal trees found during the dynamic programming algorithms used by ASTRAL and FastRFS, but SIESTA is generalizable to any algorithm that uses the same basic dynamic programming structure. Once the data structure is computed, it can be used in multiple ways to explore the solution space. In particular, it can be used to count the number of optimal solutions and determine the support for a particular bipartition, thus enabling the estimation of the support on branches for a given optimal tree that takes into account the existence of other optimal trees.

We studied SIESTA in conjunction with ASTRAL and FastRFS on a collection of biological and simulated datasets. This study showed that using SIESTA to compute the strict consensus produced a benefit for some methods in some cases, but not in all. The trends we observed clearly indicate that when there are many optimal trees, the use of the strict consensus tree results in a substantial reduction in the false positive rate and a lesser increase in the false negative rate, for an overall reduction in topological error. Conversely, when there are only a small number of optimal trees, there is little change between the strict consensus tree and any single optimal tree. Thus, the impact of using the strict consensus depends on the number of optimal solutions, which tended to be larger for all FastRFS variants than for ASTRAL. We also saw that the number of optimal trees for ASTRAL depends on the amount of missing data, so that the benefit of using SIESTA with ASTRAL to compute the strict consensus seems to be reliable only when there is missing data. The study also showed that FastRFS typically benefited from using the strict consensus tree, while ASTRAL's benefit varied with the dataset, as a result of the differences in numbers of optimal trees.

Our study showed that using SIESTA to produce a maximum clade credibility (MCC) tree with ASTRAL provided a more statistically meaningful point estimate of the true species tree than any single optimal ASTRAL tree, especially with respect to appropriately modified branch support values that take the multiple optima into account. Thus, SIESTA provides multiple benefits to species tree and supertree estimation: identifying cases where there is a unique optimum and providing better point estimates of the true tree when there are multiple optima.

Finally, there are many other methods that also use a dynamic programming approach for tree estimation (often within a constrained search space), and SIESTA can be used with these methods in similar ways. Future work should explore the impact of SIESTA with these other methods.


100 Taxa
500 Taxa
1000 Taxa
FastRFS$_{enh}$
FastRFS$_{enh}$-strict
Error
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
20%
50%
75%
100%
scaffold

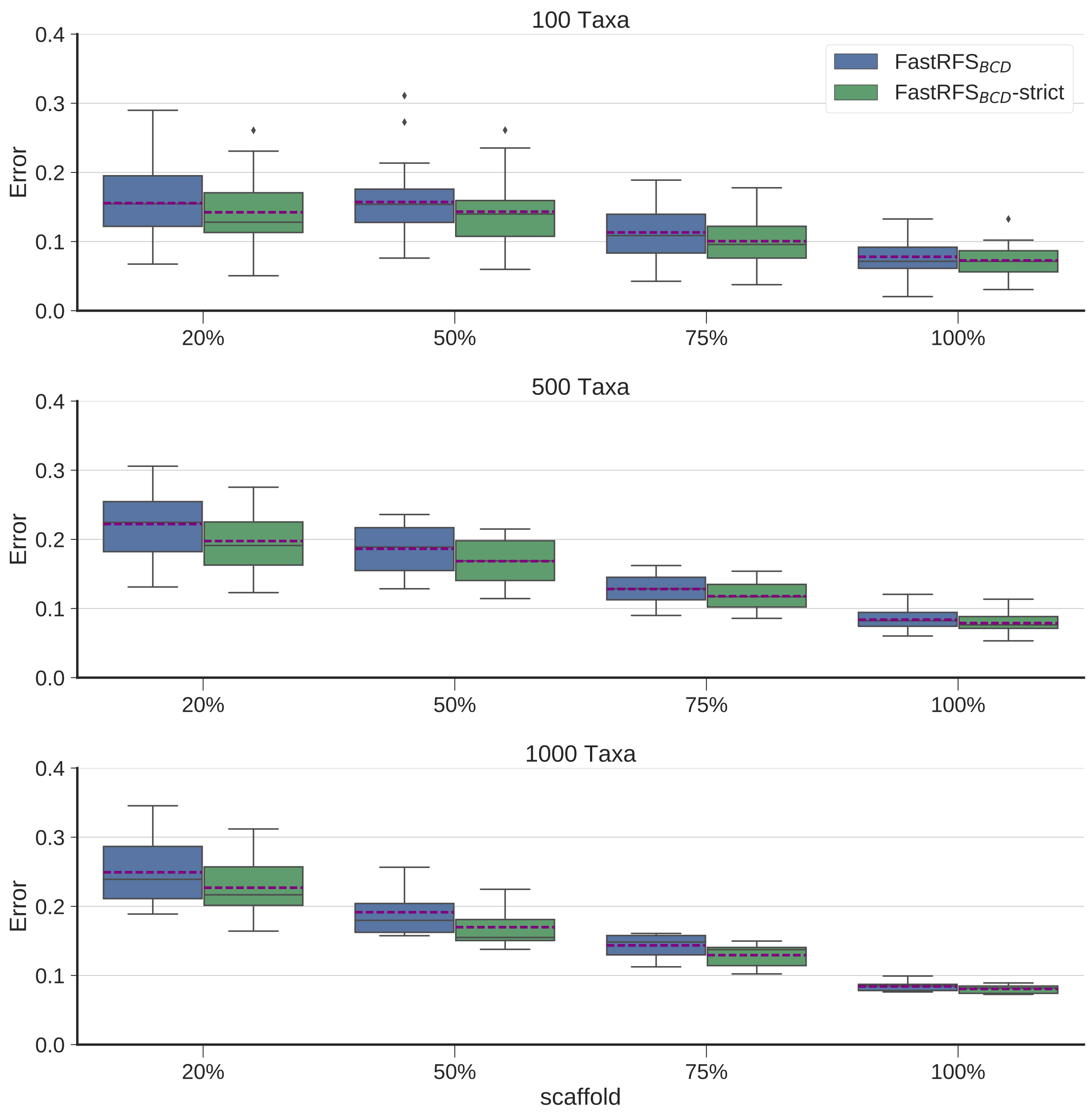


Figure 4.2: We compare a single FastRFS$_{BCD}$ supertree (FastRFS$_{BCD}$) to FastRFS$_{BCD}$+SIESTA (the strict consensus of the optimal FastRFS$_{BCD}$ supertrees) on rooted supertree datasets. Error shown is the normalized average topological error (i.e., average of FN and FP rates) between true and estimated supertrees. Error bars indicate the standard error. There are 25 replicates each for the 100- and 500-taxon datasets, and 10 replicates for the 1000-taxon datasets.

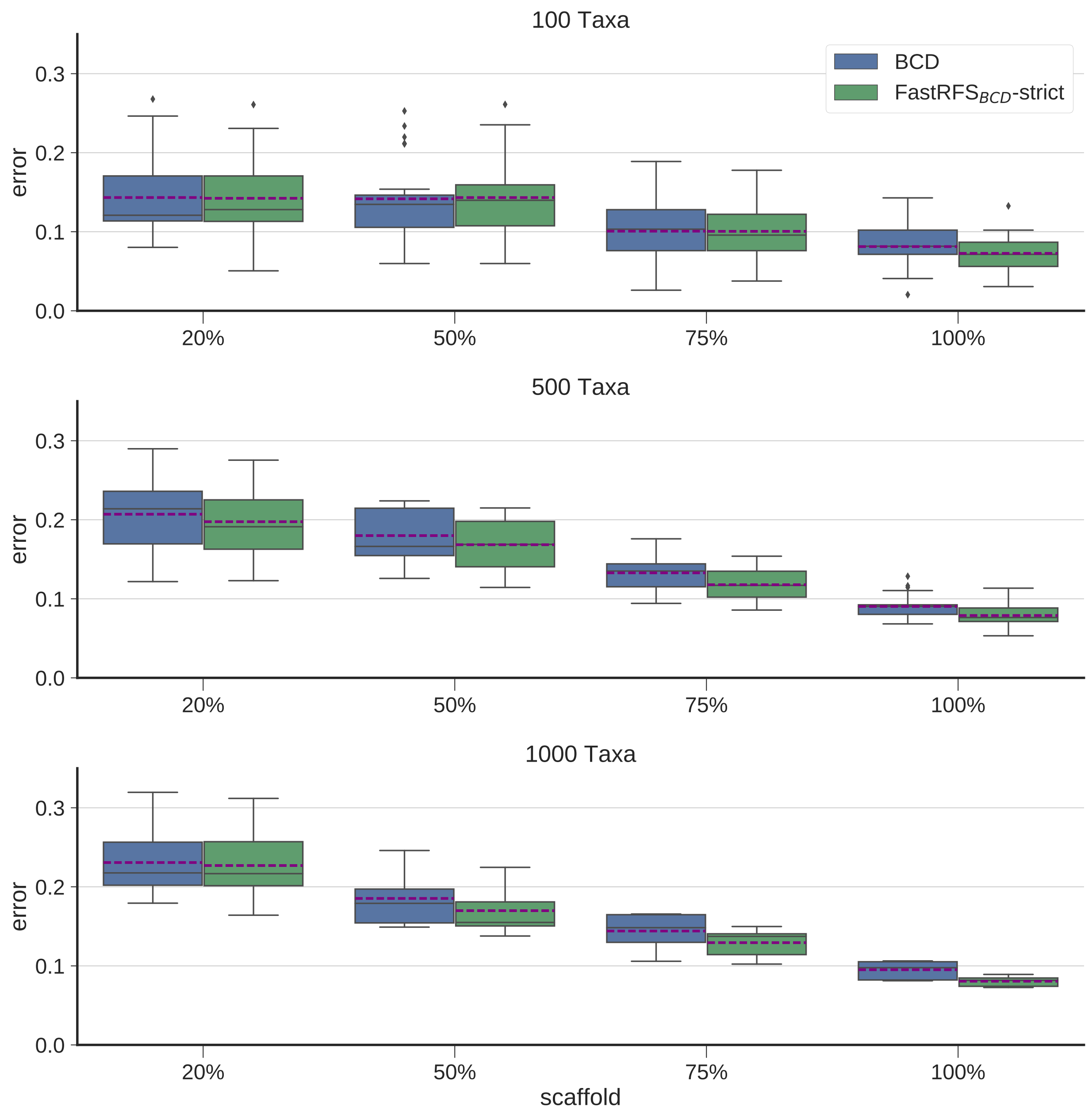


Figure 4.3: We compare Bad Clade Deletion (BCD) supertrees to the strict consensus of FastRFS$_{BCD}$ supertrees on rooted supertree datasets. Error shown is the normalized average topological error (i.e., average of FN and FP rates) between true and estimated supertrees. Error bars indicate the standard error. There are 25 replicates each for the 100- and 500-taxon datasets, and 10 replicates for the 1000-taxon datasets.

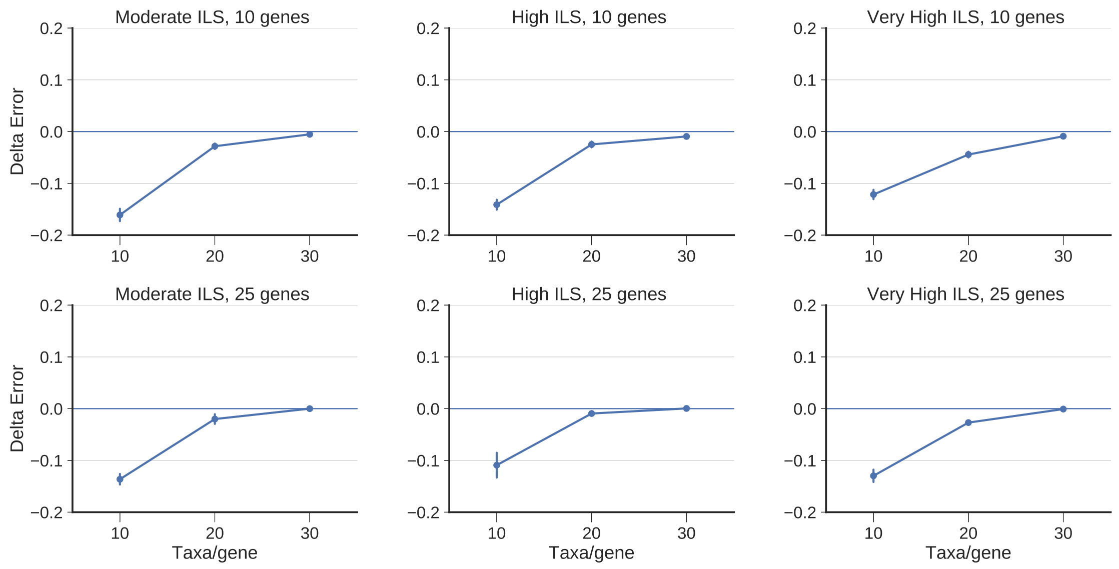


Figure 4.4: We show Delta-error (change in mean topological error between a single ASTRAL tree and the strict consensus of the set of ASTRAL trees) on simulated phylogenomic datasets with three different ILS levels, 50 species, and 25 incomplete estimated gene trees; values below 0 indicate that the strict consensus of the ASTRAL trees is more accurate than a single ASTRAL tree. We show results for 25 replicates. Error bars indicate the standard error; topological error is the average of the FN and FP error rates.

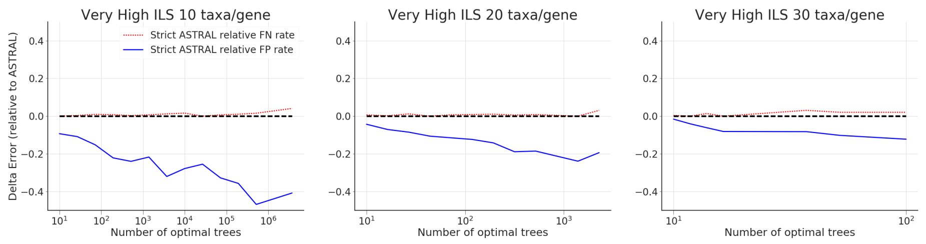


Figure 4.5: We show the FN and FP error rates of the strict consensus of ASTRAL trees, compared to a single ASTRAL tree, on simulated phylogenomic datasets with 50 species and 25 incomplete estimated gene trees; values below 0 indicate that the strict consensus ASTRAL is more accurate for that criterion (i.e., it has lower error) than ASTRAL. The x-axis shows the number of optimal trees, and we show results for 25 replicates. Error bars indicate the standard error.

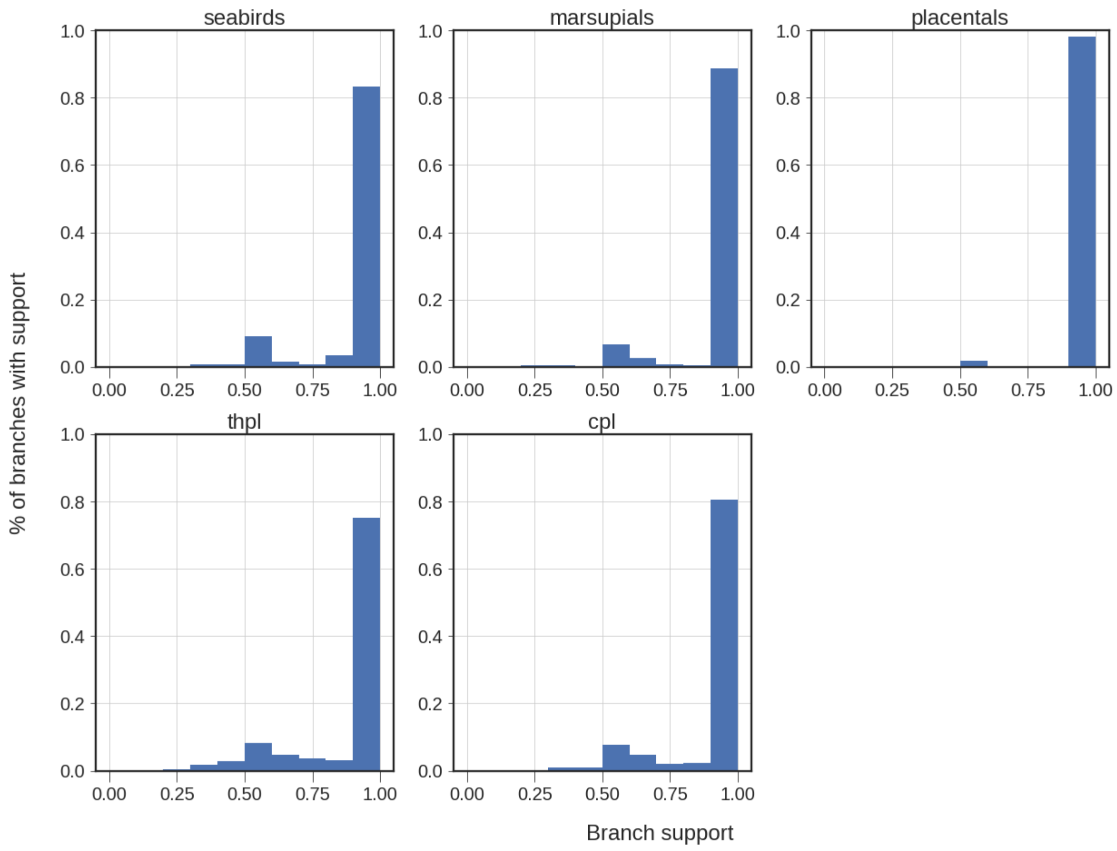


Figure 4.6: Histogram of support values for edges in the FastRFS$_{enh}$ greedy consensus tree on the unrooted supertree datasets. These support values are the percentages of the optimal trees they appear in. Although the majority of the edges have 100% support in each tree, some edges have low support, suggesting that they are not as reliable as the higher support edges.

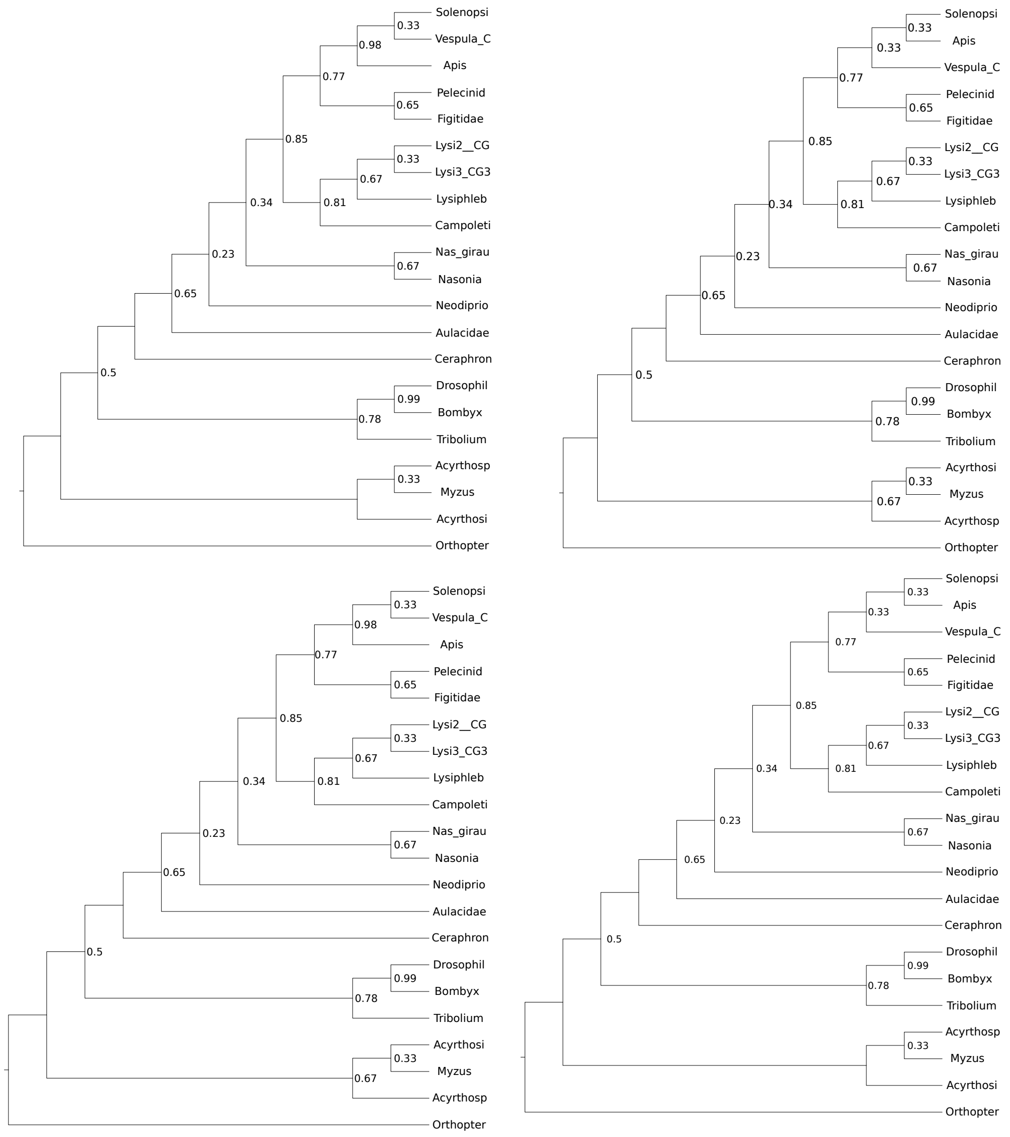


Figure 4.7: The four optimal ASTRAL trees on the Hymenoptera dataset, each rooted at the outgroup, and given with local posterior probabilities for branch support. The four trees differ only in two groups: (1) Solenopsi, Apis, and Vesputal_C, and (2) Acyrthosi, Myzus, and Acyrthosp.

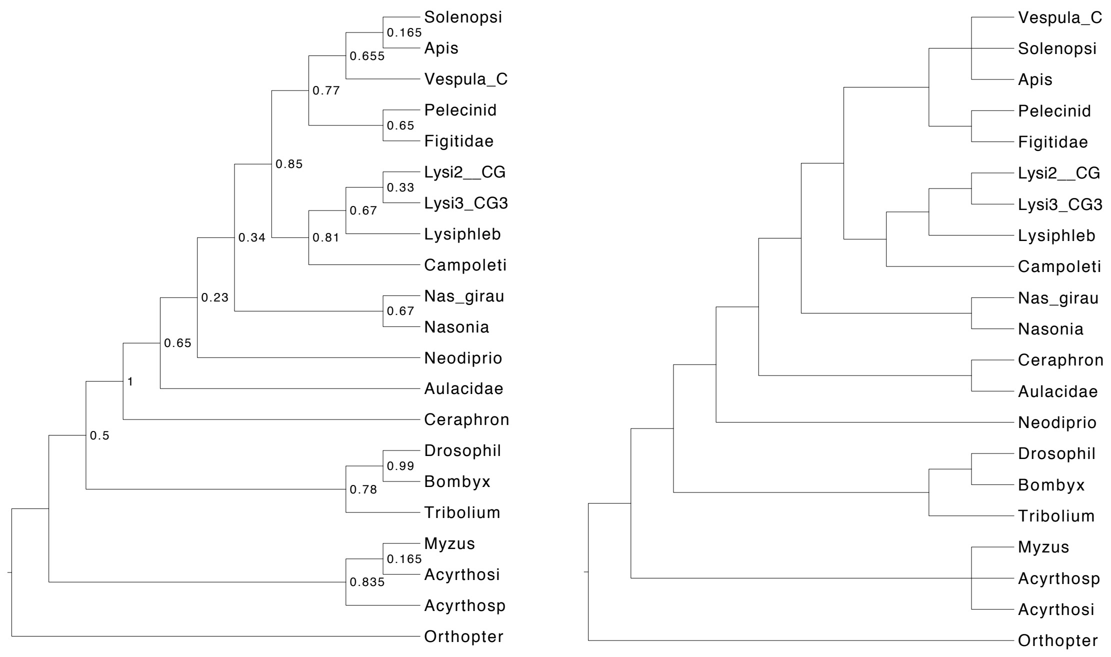


Figure 4.8: The ASTRAL Maximum Clade Credibility (MCC) tree (left) with branch support and the strict consensus tree (right) on the Hymenoptera dataset. The ASTRAL MCC tree is topologically identical to one of the four ASTRAL trees, but has different branch support; in particular, the branch support on the clades in question is half the branch support in the original ASTRAL trees on these clades. The ASTRAL strict consensus tree makes these two clades into polytomies.

4.5 Tables

Table 4.1: Statistics for biological supertree datasets. We show the number of taxa, source trees, and FastRFS$_{enh}$ supertrees for each supertree dataset.

Dataset	# Taxa	# Source trees	# FastRFS supertrees
Marsupials [23]	267	158	258048
Placental Mammals [15]	116	726	4
Seabirds [56]	121	7	117760
THPL [152]	558	19	5.9 x 10^{34}
CPL [87]	2228	39	7.7 10^{92}

Table 4.2: Statistics of the biological phylogenomic datasets. We show the number of taxa, number of genes, and number of optimal trees for ASTRAL.

Dataset (publication)	# Taxa	# Genes	# ASTRAL trees
Ferns [119]	85	25	1
Flatfishes [16]	152	23	1
Gallopheasants [90]	18	1479	1
Hymenoptera [123]	21	24	4
Lichens [68]	31	303	1
Louse [5]	15	1101	1
Mammalian [127]	37	424	1
Sigmontidine Rodents [81]	285	11	72
Skinks [70]	16	429	1
Synchaeta [138]	32	27	2
Testudinella [138]	25	27	7

4.6 Supplementary data for SIESTA

4.7 Software Commands and Version Numbers

We provide the detailed commands for the various analyses we performed.

- RAxML v8.2.6 was used to estimate gene trees on the phylogenomic simulated data with arguments "-m GTRGAMMA -p 12345 -n <jobname> -s <input>".
- RAxML v8.2.6 was used to run MRL with arguments "-m BINGAMMA -p 12345 -n <jobname> -s <input>".

Table 4.3: Number of optimal trees found by FastRFS$_{basic}$, FastRFS$_{enh}$, and ASTRAL for biological supertree datasets.

Dataset (publication)	FastRFS$_{basic}$	FastRFS$_{enh}$	ASTRAL
Seabirds [56]	17664	117760	24
Marsupial [23]	24576	258048	96
Placental [15]	64	4	4
THPL [152]	2.7×10^{18}	5.9×10^{34}	1.1×10^{11}
CPL [87]	5.4×10^{64}	7.7×10^{92}	3.9×10^{29}

Table 4.4: Running time (in seconds, rounded to the nearest tenth) on the 1000-taxon rooted supertree datasets for FastRFS$_{basic}$ and for the computation of the strict consensus of the FastRFS$_{basic}$ optimal trees (averaged over 10 replicates). The difference in running time to compute the strict consensus of the set of optimal trees compared to computing a single best tree is at most 0.3 seconds.

Scaffold factor	FastRFS$_{basic}$ (single)	FastRFS$_{basic}$ (strict consensus)	Difference
20%	31.6	31.6	< 0.1
50%	39.3	39.4	0.1
75%	37.5	37.8	0.3
100%	34.6	34.6	< 0.1

Table 4.5: Running time (in seconds) on the 1000-taxon rooted supertree datasets for BCD, FastRFS$_{BCD}$, and for the computation of the strict consensus of the FastRFS$_{BCD}$ optimal trees (averaged over 10 replicates). The difference in running time to compute the strict consensus of the set of optimal trees compared to computing a single best tree is at most half a second.

Scaffold factor	BCD	FastRFS$_{BCD}$ (single)	FastRFS$_{BCD}$ (strict consensus)	Difference
20%	10.2	33.1	33.5	0.4
50%	8.1	41.8	42.3	0.5
75%	9.2	39.9	40.1	0.2
100%	14.4	36.3	36.4	0.1

- Mrpmatrix (available from `https://github.com/smirarab/mrpmatrix`) was used to calculate the matrix for MRL.
- ASTRAL v4.7.8 was passed to FastRFS and ASTRAL-SIESTA to calculate the search space.
- BCD v1.0.1 was used to calculate BCD trees using arguments "–filetype newick"
- FastRFS v2.0 was used with and without SIESTA, with arguments "–count", "–greedy", "–majority", "–strict", and "–single" used as necessary to count the number of optimal trees, output consensus trees, or output a single optimal tree. The "-e" option was used to pass it additional trees.
- ASTRID v1.1 was used to calculate trees for the constraint set of FastRFS-enhanced, using no additional options.
- Dendropy v4.0.3 was used to calculate error rates with the function

```
dendropy.calculate.treecompare.false_positives_and_negatives
```

4.8 Additional Figures and Tables

taxa	ngenes	scaffold	astral	fastrfs-basic	fastrfs-enh
100	6	20%	9.36	3.52×10^{2}	1.21×10^{3}
100	6	50%	4.00	1.31×10^{2}	1.71×10^{3}
100	6	75%	1.72	7.27×10^{1}	1.57×10^{2}
100	6	100%	1.04	2.49×10^{1}	3.40×10^{1}
500	16	20%	1.62×10^{3}	6.09×10^{7}	1.96×10^{9}
500	16	50%	3.94×10^{1}	1.97×10^{8}	7.62×10^{8}
500	16	75%	4.23×10^{1}	1.37×10^{8}	6.99×10^{8}
500	16	100%	1.00	5.36×10^{6}	2.93×10^{7}
1000	26	20%	6.48×10^{5}	2.32×10^{15}	2.50×10^{16}
1000	26	50%	3.60×10^{4}	9.17×10^{14}	1.11×10^{18}
1000	26	75%	5.67×10^{2}	2.51×10^{14}	1.68×10^{17}
1000	26	100%	1.00	1.97×10^{13}	5.03×10^{13}

Table 4.6: Number of FastRFS optimal trees for simulated unrooted supertree datasets. We show the mean number of optimal trees averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

taxa	ngenes	scaffold	fastrfs-basic	fastrfs-bcd	fastrfs-enh
100	6	20%	1.16×10^3	6.06×10^3	4.35×10^3
100	6	50%	5.20×10^2	1.84×10^4	5.19×10^3
100	6	75%	3.00×10^2	1.40×10^3	6.47×10^2
100	6	100%	3.86×10^1	4.95×10^1	4.33×10^1
500	16	20%	4.02×10^{14}	5.42×10^{17}	1.12×10^{16}
500	16	50%	3.57×10^{14}	1.06×10^{22}	9.19×10^{19}
500	16	75%	2.05×10^{12}	7.83×10^{14}	1.65×10^{15}
500	16	100%	4.51×10^7	1.08×10^8	6.55×10^7
1000	26	20%	2.35×10^{29}	4.08×10^{37}	1.28×10^{34}
1000	26	50%	2.80×10^{29}	5.08×10^{36}	2.58×10^{37}
1000	26	75%	2.73×10^{21}	4.27×10^{29}	4.42×10^{27}
1000	26	100%	2.06×10^{14}	4.18×10^{15}	1.54×10^{15}

Table 4.7: Number of FastRFS optimal trees for simulated rooted supertree datasets. We show the mean number of optimal trees averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

ILS	ngenes	astral
Moderate ILS	5	2.12
Moderate ILS	10	1.12
Moderate ILS	25	1.04
High ILS	5	1.64
High ILS	10	1.00
High ILS	25	1.00
Very High ILS	5	1.20
Very High ILS	10	1.08
Very High ILS	25	1.04

Table 4.8: Number of ASTRAL optimal trees for simulated 50-taxon phylogenomic datasets where all gene trees are complete (i.e., no missing data). We show the mean number of optimal trees averaged over 25 replicates

	taxa/gene	10	20	30
ILS	ngenes			
Moderate ILS	5	2.87×10^2	7.07×10^2	2.41×10^1
Moderate ILS	10	1.33×10^5	7.01×10^2	1.70×10^1
Moderate ILS	25	1.80×10^7	4.68×10^1	1.80
High ILS	5	1.71×10^2	2.10×10^2	1.55×10^1
High ILS	10	8.17×10^4	6.12×10^2	1.58×10^1
High ILS	25	2.79×10^5	1.03×10^1	1.44
Very High ILS	5	1.76×10^2	1.55×10^2	1.22×10^1
Very High ILS	10	1.67×10^4	1.92×10^2	3.64
Very High ILS	25	1.08×10^5	2.42×10^1	1.40

Table 4.9: Number of ASTRAL optimal trees for simulated 50-taxon phylogenomic datasets with missing data (i.e., gene trees can be incomplete). We show the mean number of optimal trees averaged over 25 replicates for each model condition.

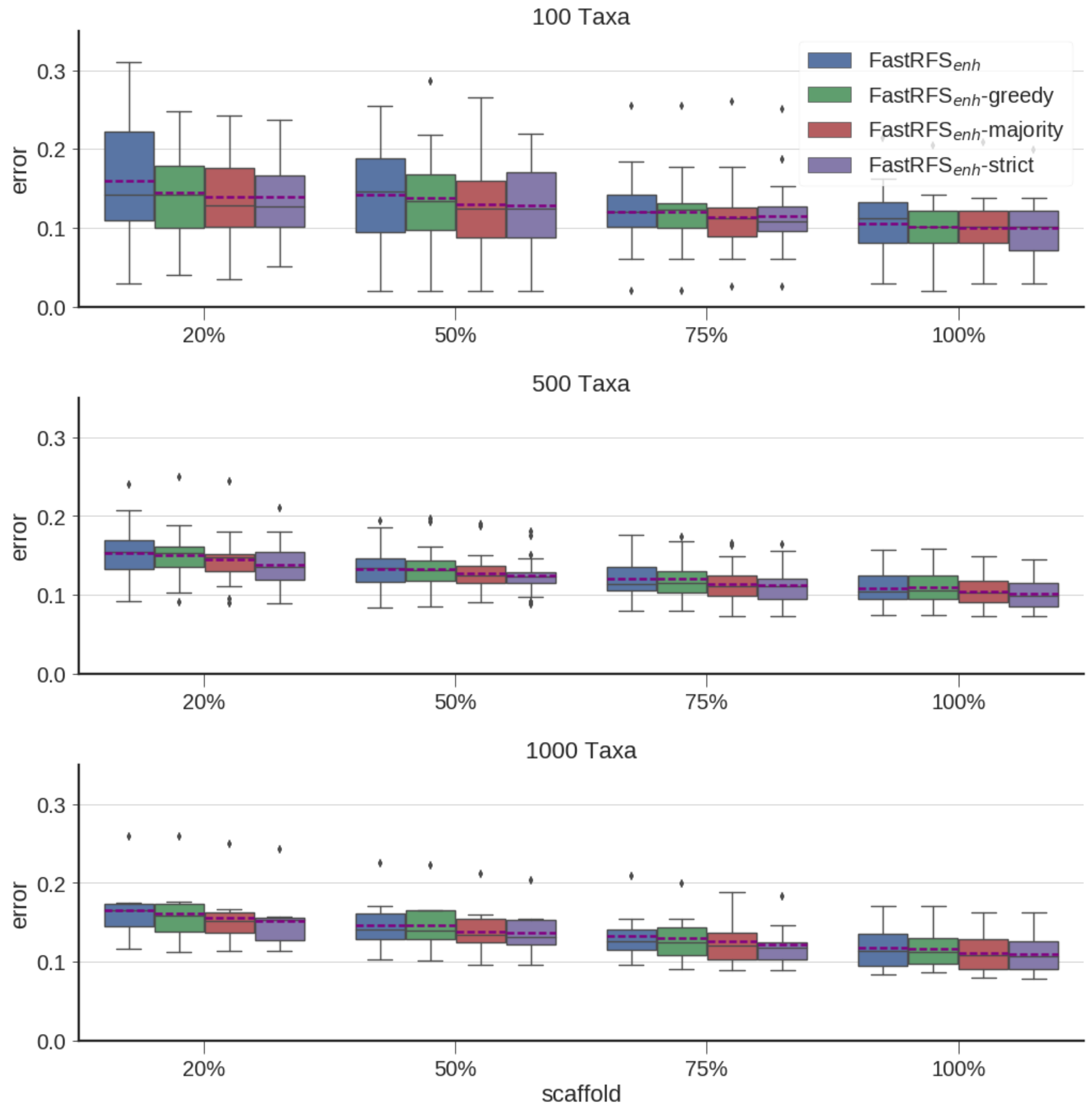


Figure 4.9: Comparison of average of FP and FN error rates for a single FastRFS$_{enh}$ tree as well as the three consensus trees computed on the optimal FastRFS$_{enh}$ trees on simulated unrooted supertree datasets. We show the mean number of optimal trees averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

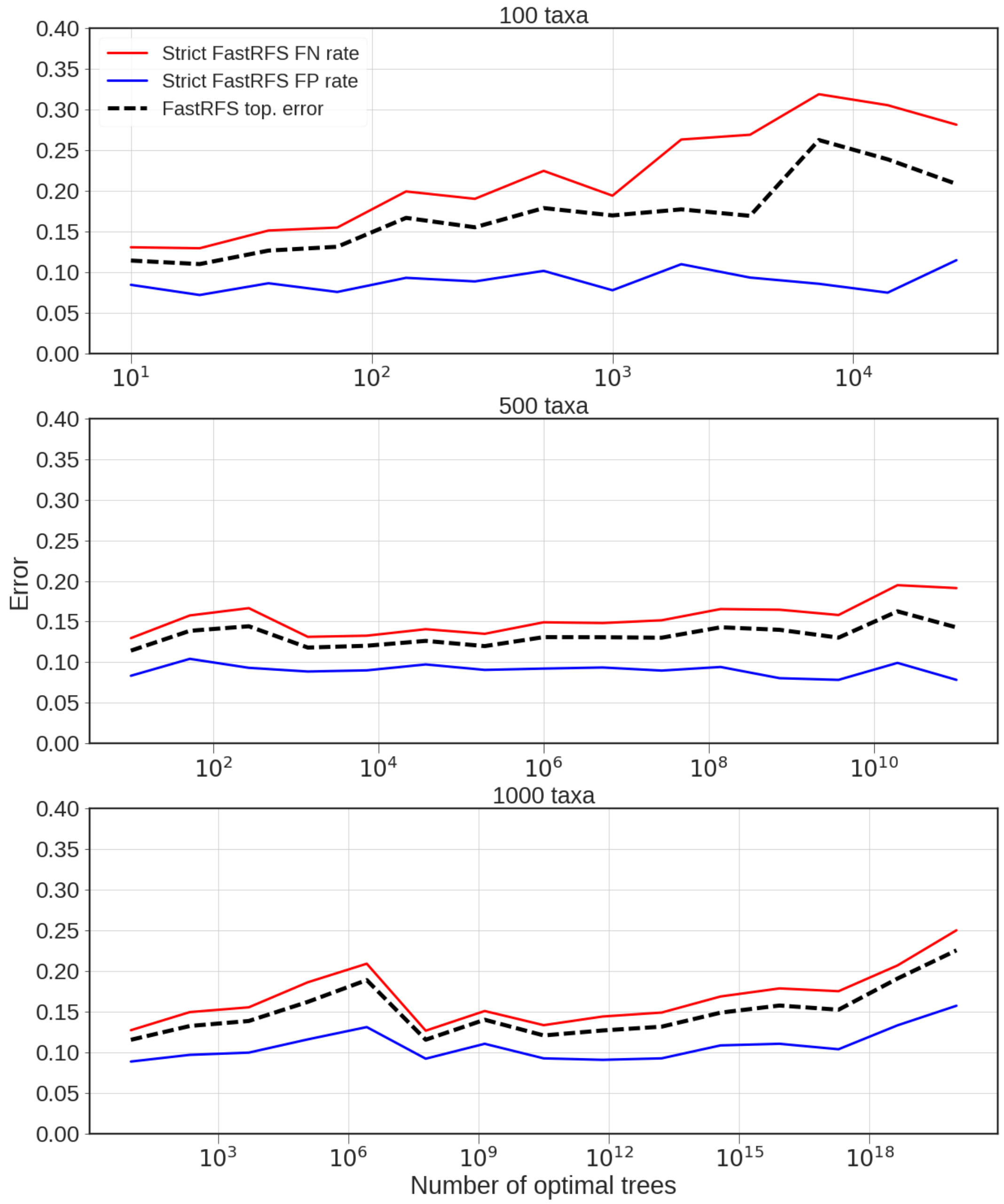


Figure 4.10: FP and FN rates for FastRFS$_{enh}$ on simulated unrooted supertree datasets as a function of the number of optimal trees. Data gathered from 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa. Red curves show false negative rates; blue curves show false positive rates.

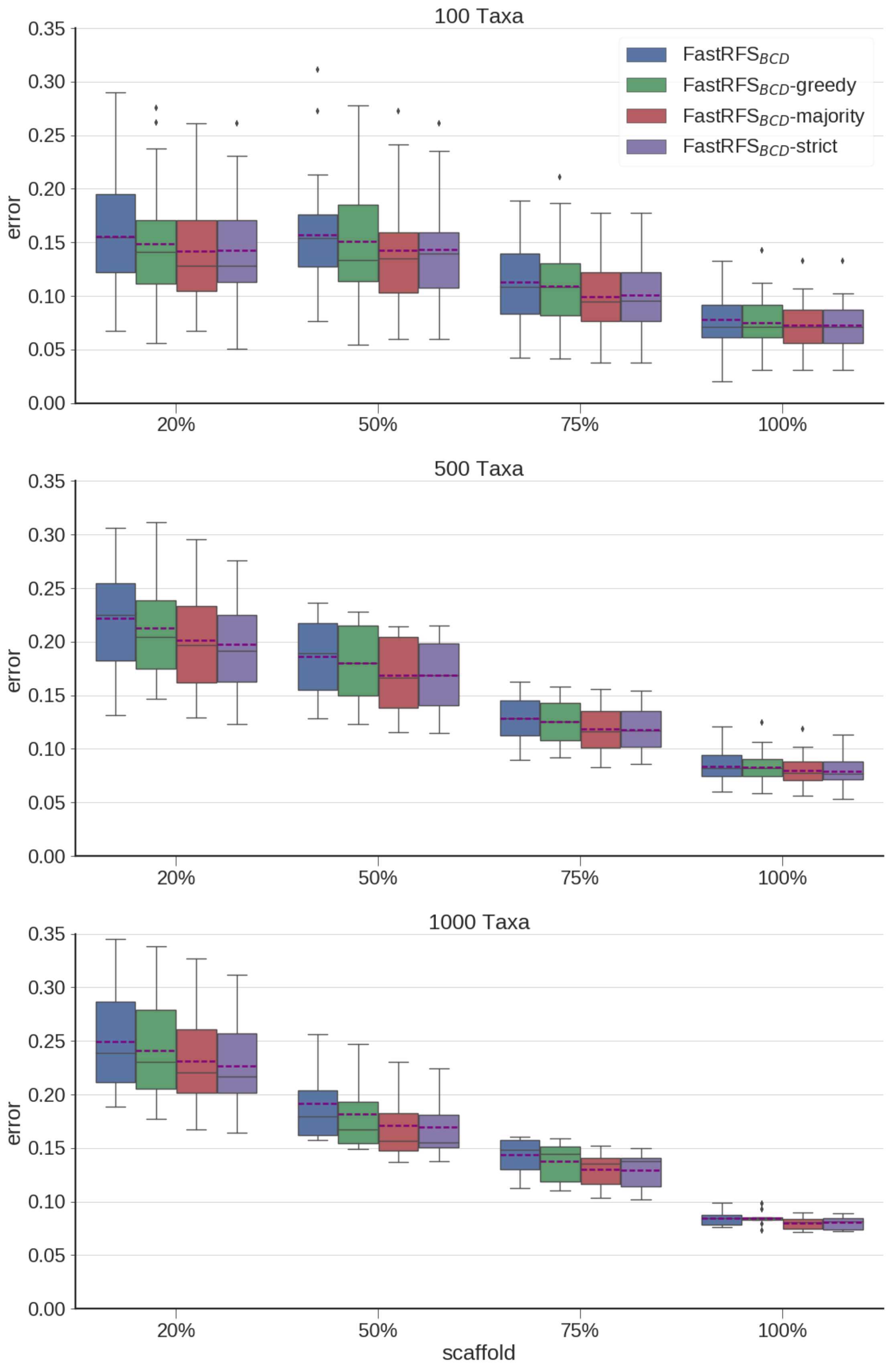


Figure 4.11: Comparison of average FP and FN error rates for a single best FastRFS$_{BCD}$ tree and three consensus trees of the best FastRFS$_{BCD}$ trees on simulated rooted supertree datasets. We show the mean error averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

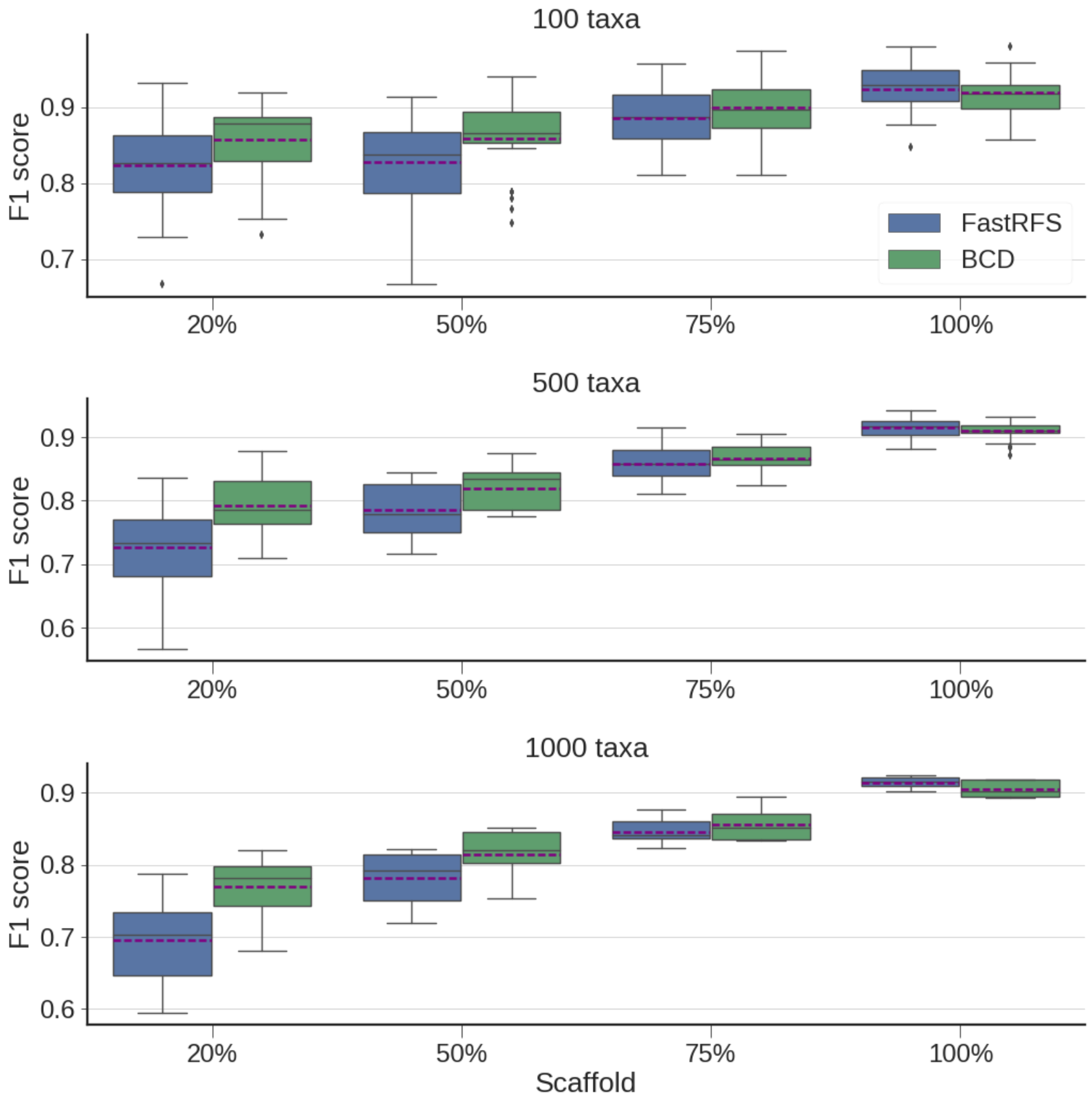


Figure 4.12: Comparison of F1 scores for a single best FastRFS-basic tree and BCD on simulated rooted supertree datasets. We show the mean scores averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

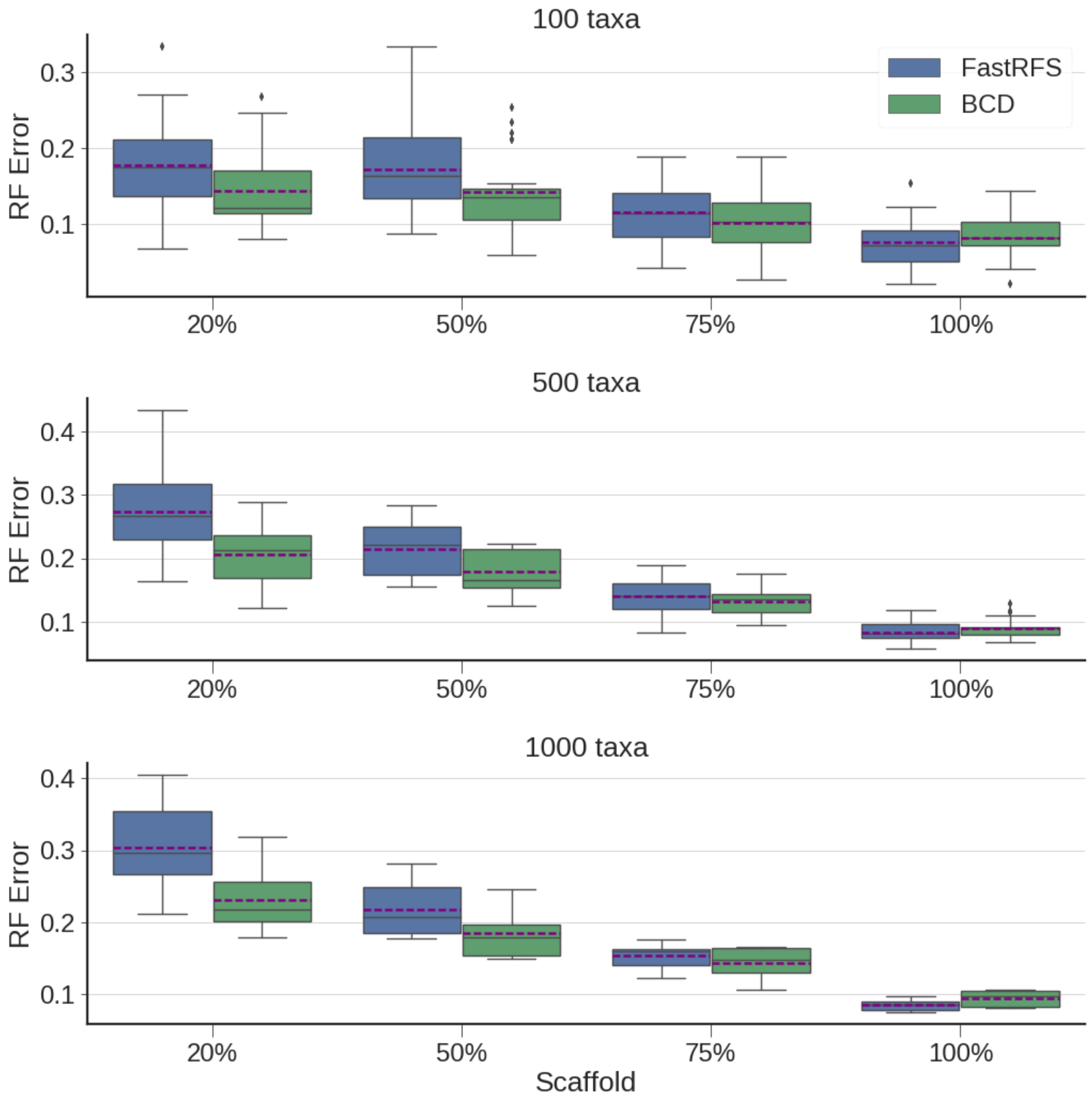


Figure 4.13: Comparison of average RF error rates for a single best FastRFS-basic tree and BCD on simulated rooted supertree datasets. We show the mean error averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

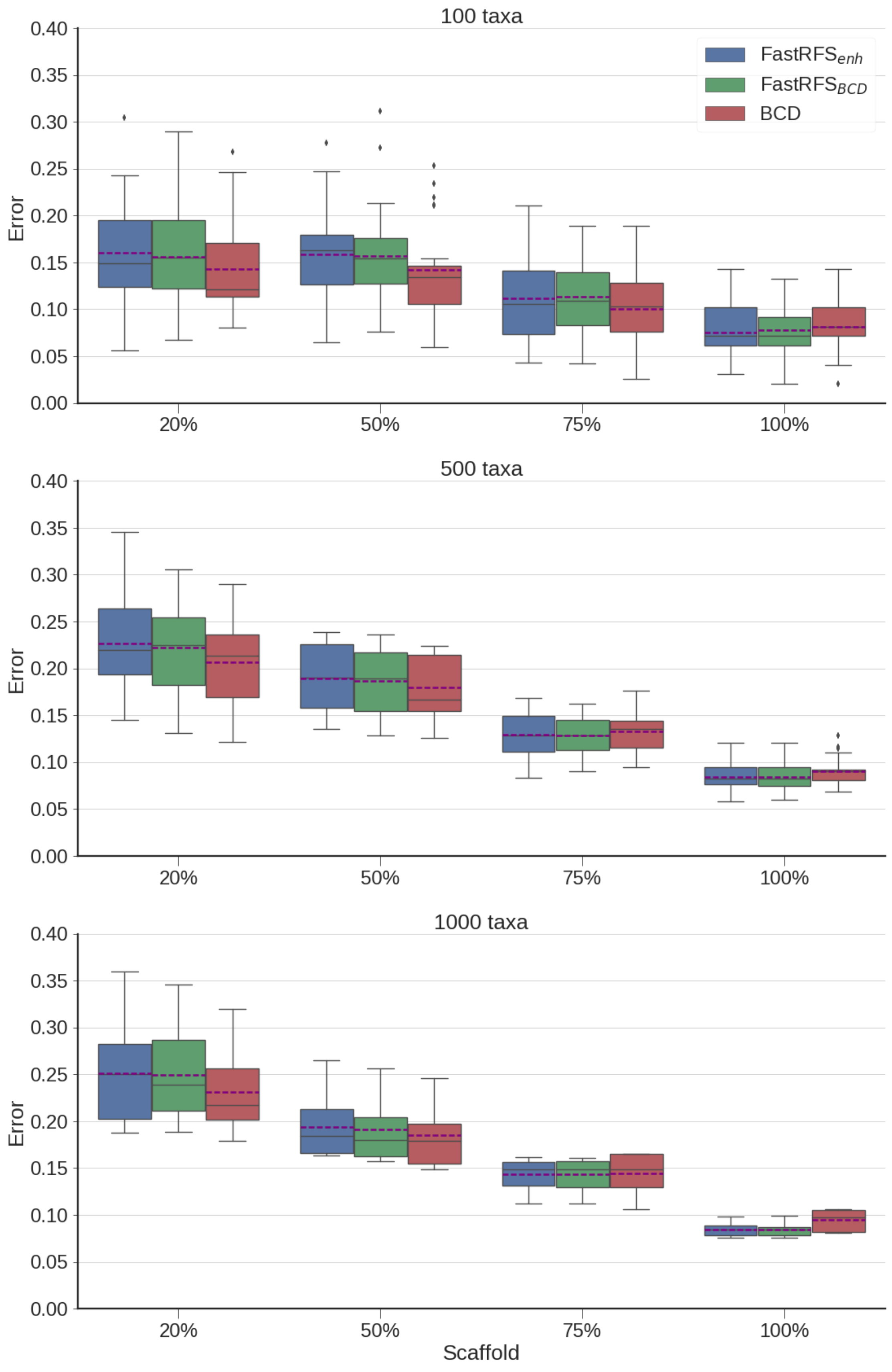


Figure 4.14: Comparison of average RF error rates for BCD and the single best trees for FastRFS$_{BCD}$ and FastRFS$_{enh}$ on simulated rooted supertree datasets. We show the mean error averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

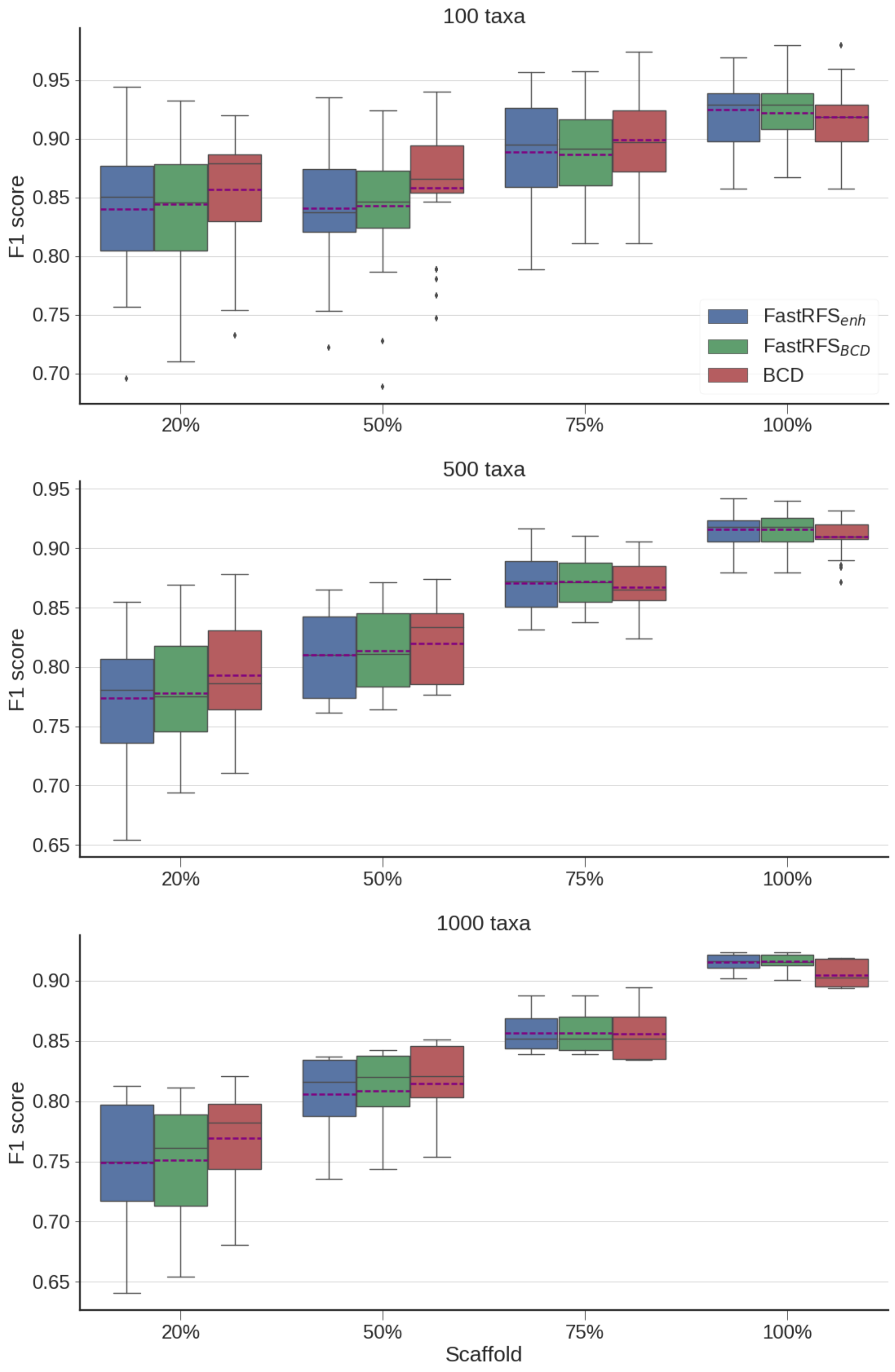


Figure 4.15: Comparison of F1 scores for BCD and the single best trees for FastRFS$_{BCD}$ and FastRFS$_{enh}$ on simulated rooted supertree datasets. We show the mean F1 score averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa.

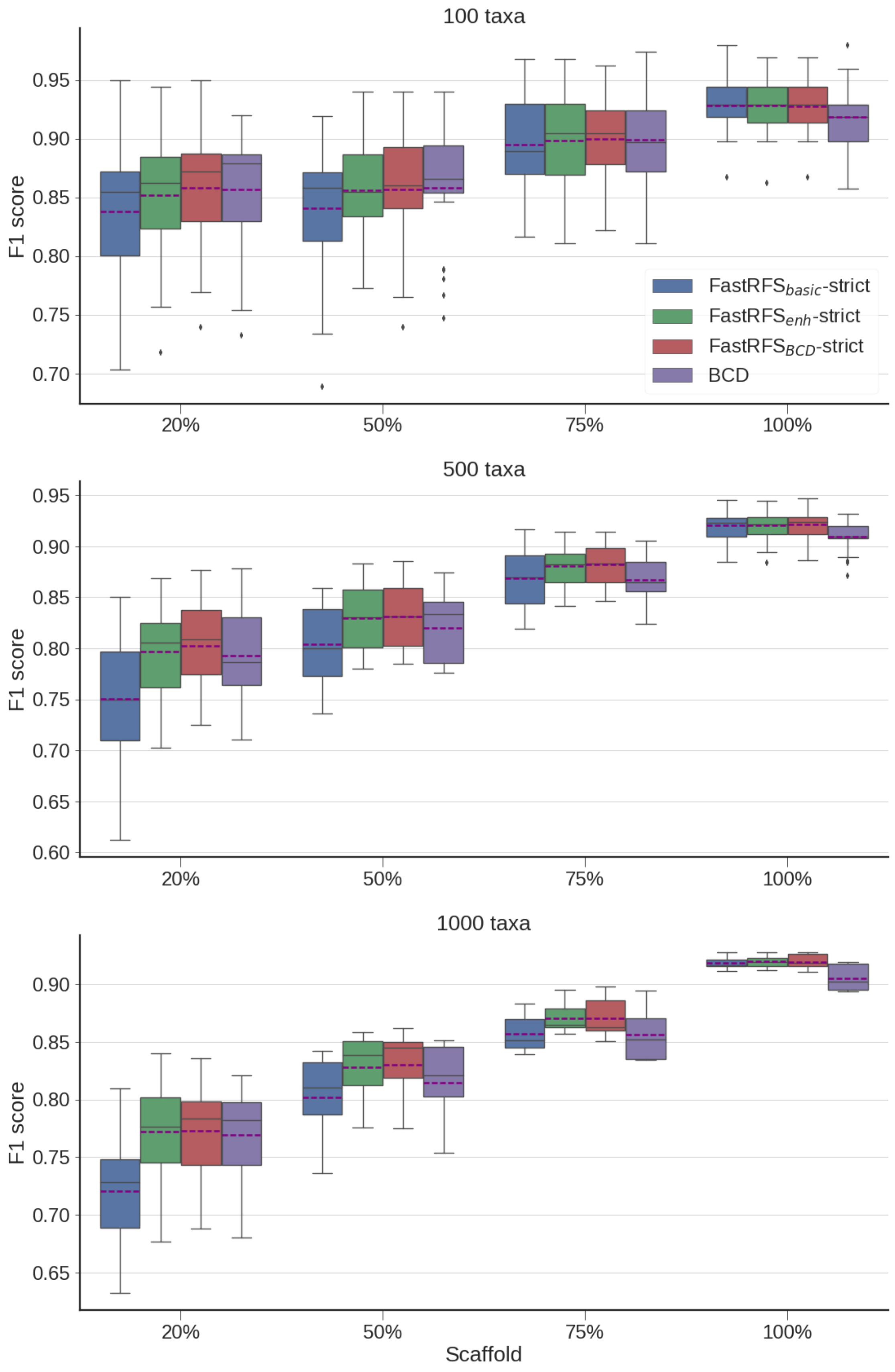


Figure 4.16: Comparison of F1 scores for the strict consensus of the optimal FastRFS$_{BCD}$ trees and BCD on simulated rooted supertree datasets. We show the mean scores averaged over 25 replicates for 100 and 500 taxa, and 10 replicates for 1000 taxa. Pranjal, restrict to just fastrfs-bcd-strict and bcd.

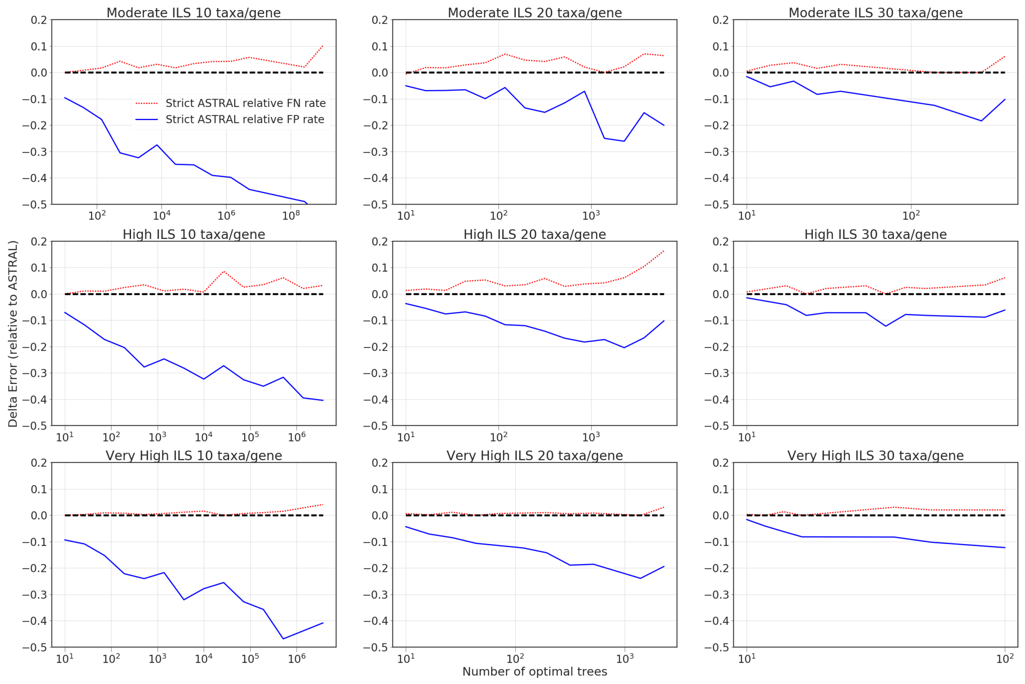


Figure 4.17: Change in FP and FN rates for the strict consensus of the optimal ASTRAL trees, compared to a single optimal tree, on simulated phylogenomic datasets as a function of the number of optimal trees. Positive values indicate that the strict consensus has a higher error than a single best tree, and negative values indicate that the strict consensus has a lower error than a single best tree. Data are gathered from 25 replicates per model condition. Red curves show false negative rates; blue curves show false positive rates.

Chapter 5

Supertree Estimation with ASTRID

5.1 Introduction

5.1.1 Supertree Estimation

Supertree estimation is an important phylogenetic problem. Methods for supertree estimation take as input a set of source trees and output a single supertree. Each input tree is on a subset of the entire taxon set, and discrepancies between the input trees result only from estimation error, not from sources of true tree heterogeneity like incomplete lineage sorting or horizontal gene transfer [147]. Supertree methods are commonly used in phylogenetic analyses (e.g. [27, 56, 88, 153]), and development of supertree methods is an area of active research [37, 105, 134, 144].

Supertree methods are used in two contexts: traditionally, they are used to combine species trees from separate analyses into one large tree. They can also be used to scale up species tree analyses on multi-locus data with gene tree heterogeneity due to incomplete lineage sorting (ILS) [80] when used as a component of divide and conquer methods [104, 146]. In these methods, a large species tree dataset is split into several small overlapping taxon sets. An accurate species tree estimation method is run on each of these small subsets, and then a supertree method is used to combine the subsets into a tree on the entire taxon set. Then, this process is repeated, with the tree from the previous iteration used to guide the new decomposition of the taxon set into overlapping subsets. Iterative divide and conquer methods enable the use of relatively slow methods, including computationally expensive Bayesian methods, on large phylogenomic datasets.

Upcoming cutting edge phylogenomic analyses will involve tens of thousands of taxa [60, 157]. Computing trees on these datasets will require scaling accurate methods beyond what is currently possible, and divide and conquer methods are a promising technique for this. However, these require supertree methods that can run on datasets with tens of thousands of taxa.

Three different kinds of methods are currently used for supertree estimation [147]. Bipartition-based methods, including Matrix Representation with Parsimony (MRP) [110], Matrix Representation with Likelihood (MRL) [105], Bad Clade Deletion [37], Robinson-Foulds supertrees [9] (such as FastRFS [144], PluMiST [63], and MulRF [24]), and likelihood-based supertrees [131], use the distribution of bipartitions in the source trees to estimate a supertree.

Quartet-based methods, most notably ASTRAL [94, 95, 154, 156], are typically used for species tree estimation from multi-locus gene trees with heterogeneity due to ILS, but can also be used for supertree estimation. They find a tree that maximizes the number of quartets (induced four-leaf trees) shared with the input trees.

Distance-based methods, including neighbor joining [120], FastME [69], and ASTRID [141], are a third type of phylogenetic estimation method that can be used for supertrees. These methods construct a distance matrix (i.e., a mapping from pairs of taxa to distances) or set of distance matrices from the input trees and construct a tree that has a distance matrix similar to the one from the input trees.

5.1.2 ASTRID

ASTRID [141] is a method for phylogenetic estimation originally based on NJst [74] and developed as a species tree method. It calculates a distance matrix from each input tree, averages them together, and uses a distance-based estimation method like neighbor joining with BIONJ* [28] or minimum-evolution estimation with FastME [69] to compute a tree. It is a statistically consistent estimation method under the coalescent model [80], on datasets with gene tree heterogeneity due to ILS.

Typically, ASTRID uses the minimum-evolution distance method FastME to estimate the species tree. FastME is fast and gives accurate results in practice. However, it requires a distance estimate for each pair of taxa; in other words, a complete distance matrix without any missing entries. If the distance matrix has missing entries, the original version of ASTRID used a neighbor joining variant called BIONJ* to estimate the species tree from the distance matrix. In practice, this is slow and tends to produce inaccurate trees.

In this paper, we present and test a new iterative approach to ASTRID when the distance matrix is missing entries. We evaluate this on biological and simulated datasets, including a very large simulated dataset with over 40,000 taxa.

5.2 Methods

Previous versions of ASTRID used BIONJ* as a distance based species tree estimation method on datasets where ASTRID's distance method was missing entries.

BIONJ* is a modified version of neighbor joining. BIONJ [40] applies a weighting scheme to the neighbor

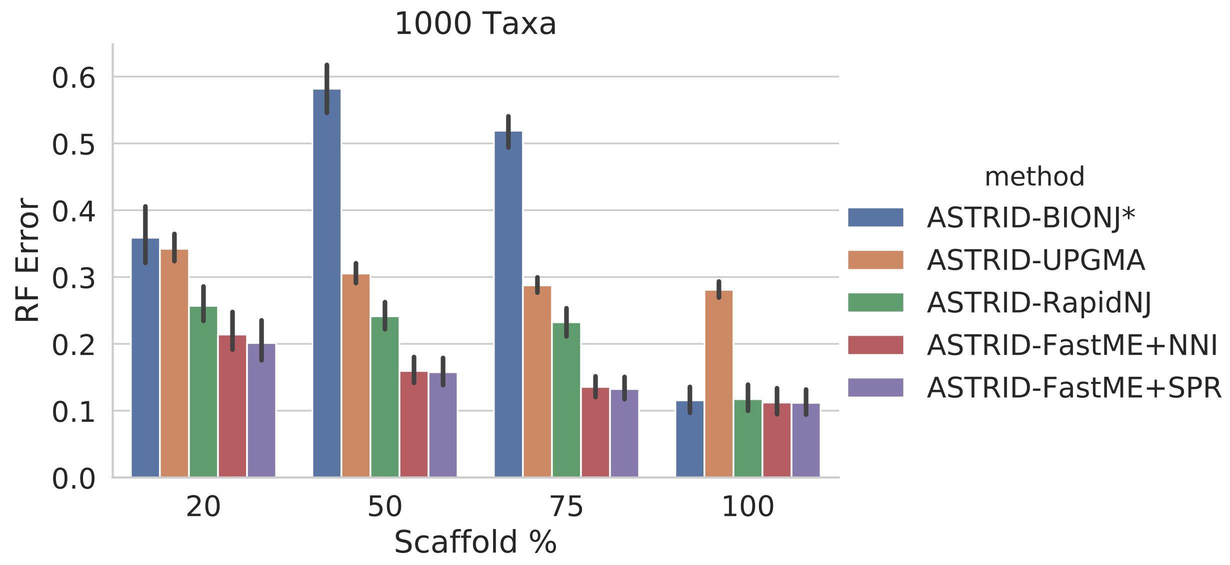


Figure 5.1: Comparison of RF errors for ASTRID with BIONJ* and ASTRID with FastME using the UPGMA protocol. ASTRID-BIONJ* is used by the original version of ASTRID when the distance matrix is missing data. Results are shown on 1000-taxon SMIDgen data with 16 source trees. Data is averaged over 10 replicates.

joining algorithm based on a simple model of variances in the distance matrix. This model, derived from the Jukes-Cantor model of sequence evolution [55], predicts that the variance of a distance estimate between two taxa is proportional to the distance between those taxa. BIONJ* further extends BIONJ by allowing for distance matrices with missing taxa. This technique combines some straightforward modifications of the BIONJ algorithm with a set of heuristics to break ties at decision points within the algorithm.

This approach suffers from several issues. First, in some cases, the accuracy suffers substantially on these datasets when BIONJ* must be used. Second, the running time of BIONJ* is $O(n^3)$ in the number of taxa, and dominates the running time of ASTRID on larger datasets with high levels of missing data. Third, and somewhat less importantly, BIONJ* requires Java to be installed and configured appropriately on the user's computer, which can be challenging for the user.

The improved version of ASTRID presented here retains the option to use BIONJ*, but adds a new iterative approach that mitigates all three of these issues. First, ASTRID uses a variant of UPGMA [126], UPGMA*, which we have developed for this purpose, to quickly generate a (likely inaccurate) species tree. Then, it uses the UPGMA* tree to fill in the missing entries in the distance matrix, allowing for a fast and accurate method that requires a complete distance matrix to be run.

5.2.1 UPGMA*

We have developed a novel distance method, UPGMA*, which extends UPGMA so that it can be used on incomplete distance matrices.

UPGMA and UPGMA* take as input an n element taxon set S and an $n \times n$ distance matrix D. They output a species tree T. UPGMA operates as follows:

1. Initialize a forest data structure F with n independent elements corresponding to the n taxa
2. Initialize n min-heap priority queues $Q_1, \ldots, Q_n$ that hold $< distance, taxon >$ pairs.
3. Initialize a map M with $\langle taxon, taxon \rangle$ pairs as the keys and *distance*s as the values.
4. Mark each taxon as alive, and set the size of each taxon to 1
5. For each pair of taxa i, j, add $\langle D[i,j], j \rangle$ to Q_i and $\langle D[i,j], i \rangle$ to Q_j
6. For each pair of taxa i, j, add $\langle i, j \rangle \to D[i,j]$ to M
7. While F is disconnected:
    - (a) For each priority queue Q_i where i is marked alive:
        - i. Pop elements from Q_i until the top element has a taxon marked alive
        - ii. Let $best_i$ be the top element of Q_i
    - (b) Let Q_x be the priority queue with the smallest top element. That top element is $best_x = \langle d_{min}, y \rangle$
    - (c) Create a new taxon xy and mark it alive. Let $size(xy) = size(x) + size(y)$
    - (d) Mark taxa x and y as dead
    - (e) Add xy to F and set its children as x and y
    - (f) Create a new priority queue Q_{xy}.
    - (g) For each alive taxon i:
        - i. Let $d_{xy,i} = \frac{size(x)M[x,i]+size(y)M[y,i]}{size(x)+size(y)}$.
        - ii. Add $\langle xy, i \rangle \to d_{xy,i}$ to M
        - iii. Add $\langle d_{xy,i}, i \rangle$ to Q_{xy}.
8. Let T be the sole tree in F. Return T.

The overall asymptotic running time for this algorithm is $O(n^2 \log n)$: in each of the $O(n)$ iterations, $O(n)$ items are added to priority queues. Each addition takes $O(\log n)$ time. Since each item can be removed from a priority queue at most once, the total time taken for both additions and removals from priority queues is $O(n^2 \log n)$.

UPGMA* differs in a few ways that allow for distance matrices to contain missing elements.

First, when the data structures are initialized, only taxon pairs where the distances are known are considered.

Second, when $d_{xy,i}$ is computed, if $M[x, i]$ is unknown, $d_{xy,i} = M[y, i]$, and if $M[y, i]$ is unknown, $d_{xy,i} = M[x, y]$. If both $M[x, i]$ and $M[y, i]$ are unknown, $d_{xy,i}$ is unknown, and the corresponding elements are not added to M or Q_{xy}.

Third, if after removing dead elements, every priority queue is empty (and the forest is still disconnected), the distance matrix is disconnected; i.e. there are two disjoint subsets of taxa S_1 and S_2 such that $S_1 \cup S_2 = S$ and $\forall_{i \in S_1, j \in S_2}, D[i, j]$ is missing. In this case, UPGMA* picks an arbitrary pair of taxa to join and outputs a warning to the user.

These changes do not increase the asymptotic running time of UPGMA*. Since UPGMA is known to not be statistically consistent, UPGMA* is also not statistically consistent.

5.2.1.1 Iterative protocol

First, ASTRID calculates a distance matrix from the input trees. This matrix, M_0, may have missing entries. Then, UPGMA* is run, resulting in a tree T_U that has a topological distance matrix M_U. Each missing entry in M_0 is replaced with the corresponding entry in M_U to get the distance matrix M_1, which has no missing entries. Now, FastME (or another distance method) can be run on M_1 to get a tree T_1. The matrix from T_1, M_1', can be used once again to fill in the missing entries in M_0 to get M_2, and FastME can again be run on this matrix. This procedure can be iterated further, and the final tree is the output.

5.2.2 Distance Methods

ASTRID can use a variety of different distance methods to compute the species tree. The balanced minimum-evolution (BME) estimator in FastME is the default method used when the distance matrix is complete. Balanced minimum-evolution estimation has some theoretical similarities to neighbor joining [41], but FastME is able to obtain more accurate results and run more quickly than standard neighbor joining implementations. FastME uses a taxon addition strategy, in contrast with the agglomerative approach used by neighbor joining, and then can improve its tree with nearest-neighbor interchanges (NNIs) and subtree prune-and-regraft

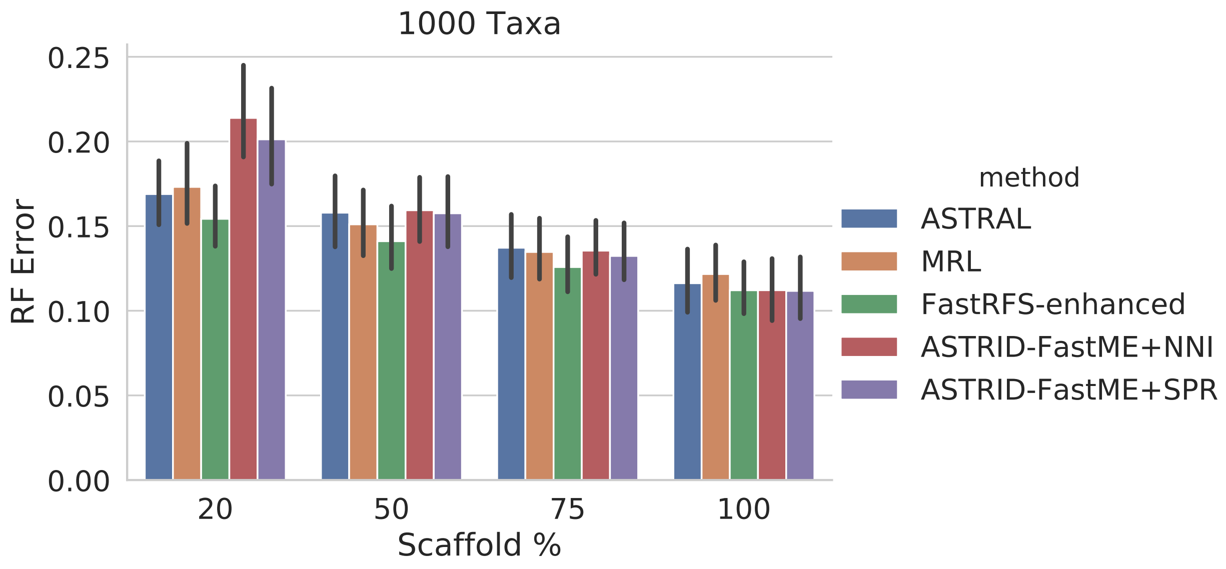


Figure 5.2: Comparison of RF errors for supertree methods and ASTRID variants with FastME using the UPGMA protocol. Results are shown on 1000-taxon SMIDgen data with 16 source trees. Data is averaged over 10 replicates.

moves (SPRs).

BIONJ*, as discussed above, is a modification to the neighbor-joining algorithm that incorporates variance estimates for the elements in the distance matrix, and also is able to run on incomplete distance matrices.

RapidNJ [125] is an implementation of neighbor joining designed to scale to extremely large datasets. It uses heuristics to significantly reduce the number of options considered in each iteration of the neighbor joining algorithm. In fact, while most distance methods require the entire distance matrix to fit into memory, RapidNJ is able to store most of the distance matrix on disk, paging portions into main memory only as needed [124]. While this has a negative impact on running time, it enables analyses of a size impossible with other methods.

5.3 Experiments

Previous studies of supertree methods [133, 144] compared the performance of leading supertree and species tree methods, including ASTRID, on simulated and biological datasets. One study [144] showed that ASTRID performed poorly in terms of accuracy and running time when the distance matrix was missing elements. For this paper, we replicated some of the analyses from those studies and also analyzed one newly simulated dataset.

5.3.1 Datasets

5.3.1.1 Simulated SMIDgen datasets

For this paper, we ran our improved version of ASTRID on SMIDgen datasets originally analyzed in [133] and also analyzed in [144]. These simulated datasets had 100, 500, or 1000 taxa. The 100 taxon replicates had 6 source trees, the 500 taxon replicates had 11 source trees, and the 1000 taxon datasets had 16 source trees. In each replicate, one source tree was a scaffold tree, which was simulated as a universal gene and contained 20%, 50%, 75%, or 100% of the taxa. The remainder of the source trees were clade-based and contained a portion of taxa from a single clade in the true model tree. The only source of heterogeneity in the source trees was due to tree estimation error. The 100 and 500 taxon datasets had 30 replicates per scaffold level, and the 1000 taxon datasets had 10, resulting in a total of 280 replicates.

5.3.1.2 Large RNAsim-based dataset

We also analyzed a very large supertree dataset derived from the RNAsim dataset originally generated for [44] and modified and used in [96]. A 30% scaffold tree was chosen by randomly sampling the taxa in a 50,000 taxon subset of the tree. Then, the following procedure was repeated:

- A random clade c with between 1% and 50% of the total taxon set was chosen
- If more than 70% of the taxa in c already existed in previously chosen clades, c was discarded
- Otherwise, 70% of the taxa in c were randomly chosen to form a clade-based tree

This process was halted when the total taxa in the clades was more than $40,000$. Ultimately, 1 scaffold tree and 30 clade-based trees were chosen with a total of $43,183$ taxa. FastTree 2.1.9 [109] was used to estimate source trees from the $21,946$ site RNA sequences with gaps.

5.3.1.3 Biological datasets

Finally, we tested the various supertree methods on some of the biological datasets previously analyzed in [144] and [134]. In particular, we analyzed the three datasets with a scaffold of less than 100% where the distance matrix was missing taxa: the 2,228 taxon comprehensive papilinoid legume (CPL) dataset with 39 trees including a 74% scaffold tree [88]; the 558 taxon temperate herbaceous papilinoid legume (THPL) dataset with 19 trees including a 25% scaffold tree [153]; and the 121 taxon seabird dataset with 7 trees, including a 74% scaffold tree. [56].

On these biological datasets, we evaluated running time and memory usage for the various supertree methods. We did not evaluate accuracy due to the lack of a known true tree.

5.3.2 Methods

5.3.2.1 ASTRID variants

We evaluated several versions of ASTRID, which varied in the distance method used to estimate the species tree. We tested UPGMA* and BIONJ*, which work by themselves on incomplete matrices. We also tested the iterative protocol using UPGMA* as the first step, then repeating FastME one, two, or three times, and using one of three FastME variants that used different local search heuristics - either no local search (ASTRID-FastME+nosearch), NNIs (ASTRID-FastME+NNI), or NNIs and SPRs (ASTRID-FastME+SPR). We also evaluated RapidNJ as the second stage of the iterative protocol, as it may be able to scale to even larger datasets than FastME.

5.3.2.2 Coalescent-aware methods

We tested ASTRAL 5.14.2 [154], a popular coalescent-aware method that exactly optimizes quartet support over input trees within a constrained search space X that it generates based on the input trees.

5.3.2.3 Supertree methods

We tested two supertree methods. First, we tested Matrix Representation with Likelihood (MRL) [105], which computes an alignment matrix based on the bipartitions in the input trees, then uses RAxML 8.2.12 [129] to estimate a tree from that alignment. On the large RNAsim-based dataset, we used FastTree 2.1.9 [109] instead of RAxML because RAxML was too slow and memory intensive to run.

We also tested FastRFS [144], which is a supertree method that exactly minimizes the Robinson-Foulds distance to the input trees using a constraint set generated by ASTRAL 5.14.2. Since FastRFS uses the same constraint set as ASTRAL, its accuracy has increased due to improvements to ASTRAL's algorithm. FastRFS may have multiple optimal solutions, and we use SIESTA [143], which is built in to FastRFS, to report a strict consensus of all optimal trees.

5.3.3 Evaluation metrics

We report estimation error on the simulated datasets only, since there is no known true tree on the biological datasets. We use the Robinson-Foulds (RF) error between the estimated and true trees, which is the average of the branch false positive and branch false negative rates. The RF error falls between 0 and 1, with an error of 0 indicating an estimated tree identical to the true tree, and an error of 1 indicating an estimated tree that shares no bipartitions with the true tree. All methods except FastRFS return binary trees, so, the false positive rate, false negative rate, and RF error are all equal for these methods.

We also report running times and memory usage on simulated and biological datasets. We analyzed the SMIDgen datasets on Blue Waters nodes with 16 AMD Interlagos cores and 64GB RAM, and we analyzed the large RNAsim-based dataset and biological datasets on an Illinois campus cluster node with 20 Intel Ivy Bridge cores and 256 GB RAM. ASTRAL and FastME are multithreaded and were allowed to run on all cores; the remainder of the methods, including the distance matrix generation step of ASTRID, are single threaded.

		ASTRAL	MRL	FastRFS-enhanced	ASTRID UPGMA	ASTRID BIONJ*	ASTRID RapidNJ	ASTRID FastME +nosearch	ASTRID FastME +NNI
Taxa	Scaffold %								
100	20	6.3	7.7	11.1	0.0	4.6	0.0	0.0	0.0
6 trees	50	6.5	7.7	11.2	0.0	5.0	0.0	0.0	0.0
	75	6.5	8.0	11.6	0.0	6.2	0.0	0.0	0.0
	100	7.1	7.7	11.2	0.0	4.7	0.0	0.0	0.0
500	20	40.4	270.0	296.6	0.2	140.0	0.2	0.3	0.3
11 trees	50	42.6	288.0	317.7	0.2	239.0	0.2	0.4	0.4
	75	44.1	321.8	353.6	0.2	556.6	0.3	0.4	0.5
	100	46.5	290.4	322.0	0.3	504.4	0.3	0.5	0.5
1000	20	102.0	1510.4	1615.4	0.7	799.4	0.9	1.8	2.0
16 trees	50	112.6	1589.8	1708.8	0.9	1947.9	1.2	2.1	2.2
	75	117.7	1888.6	2014.1	1.1	4074.3	1.3	2.3	2.5
	100	118.5	1548.5	1676.4	1.3	4259.1	1.5	2.6	2.7

Table 5.1: Comparison of running times (in seconds) for ASTRID variants and supertree methods on SMIDgen simulated data. 100- and 500-taxon datasets had 30 replicates per model condition, and 1000-taxon datasets had 10 replicates per model condition. Time for FastRFS includes time to run MRL for the expanded search space. Some methods took less than 0.05 seconds on average on the 100-taxon datasets; these are listed as taking 0.0 seconds to complete. Calculations were run on a 16-core AMD Interlagos Blue Waters node with 64 GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

5.4 Results

5.4.1 Determining the best way to run ASTRID

The first set of experiments varied the parameters for ASTRID to determine the best way to run it. In addition to testing the previous version of ASTRID, which used BIONJ*, we tested the iterative protocol with FastME with no searches, FastME+NNI, FastME+SPR, and RapidNJ for the second stage. We tested the iterative protocol with up to three iterations of the second-stage method to determine if more iterations

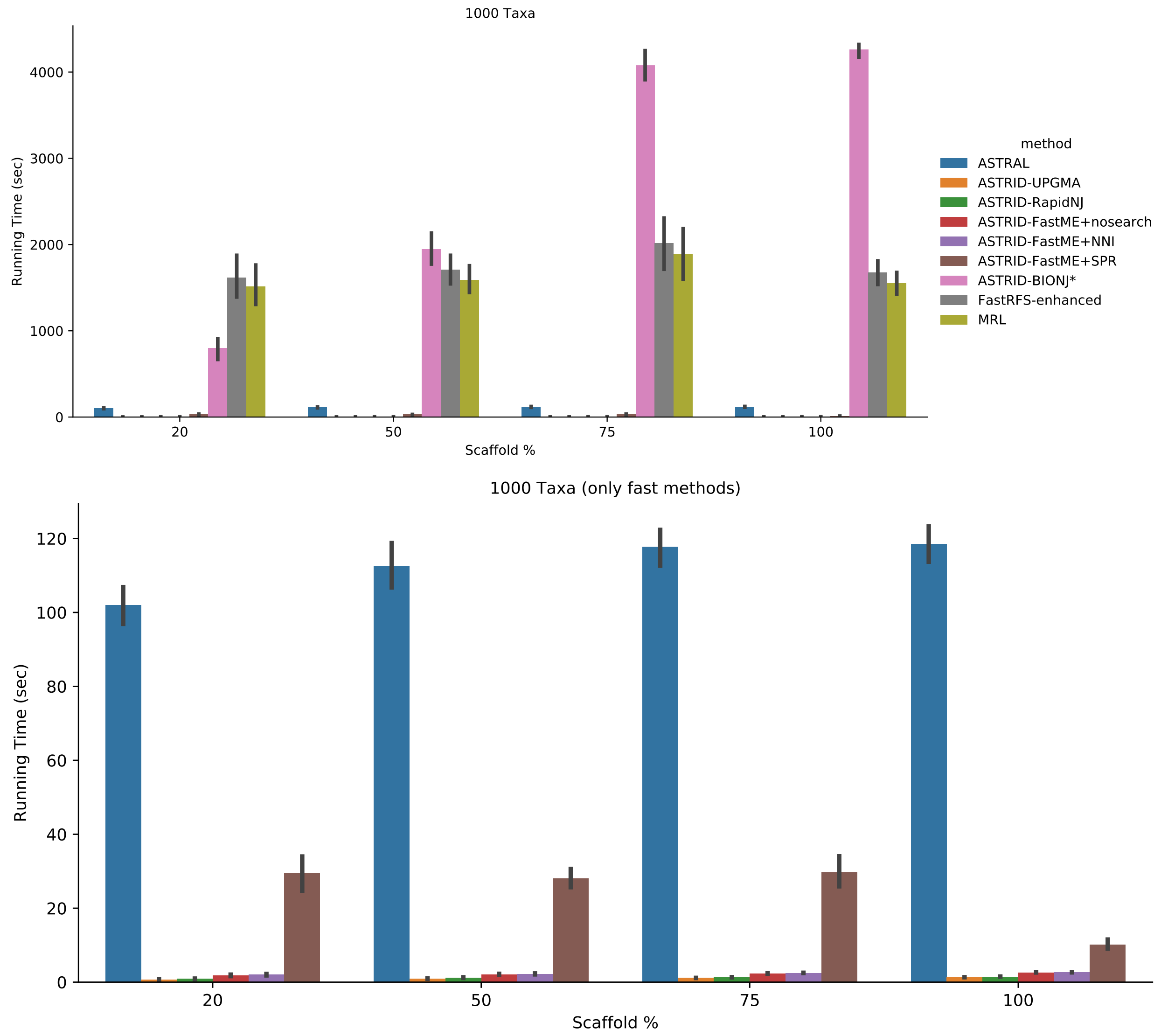


Figure 5.3: Comparison of running times for supertree methods on simulated SMIDgen data with 1000 taxa and 16 source trees. Data is averaged over 10 replicates. The second figure shows the same data, but without the slowest methods (MRL, FastRFS, ASTRID-BIONJ*). Calculations were run on a 16-core AMD Interlagos Blue Waters node with 64 GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

improved the results.

5.4.1.1 Number of iterations

We first compare running multiple iterations of FastME with no local search, NNIs, or NNIs and SPRs (see Fig. 5.4).

These experiments show that additional iterations of FastME beyond the first do not improve the results for any of the FastME search types; in fact, there was never any difference between the number of iterations.

However, there was a significant extra running time cost for additional iterations, especially for ASTRID-FastME+SPR, where the running time was dominated by the cost to run the distance method.

Therefore, for the remainder of the experiments, we only used a single iteration of the second stage distance methods.

5.4.1.2 FastME local search variants

FastME with no local search is typically less accurate than using NNIs and SPRs (see Fig. 5.4). FastME+SPR is always at least as accurate as FastME+NNI, and in some cases is slightly more accurate (see Fig. 5.1). This difference was most pronounced on the 20% scaffold datasets.

NNIs take essentially zero additional time, but SPRs come at a significant running time cost, as seen in Table 5.1 and Fig. 5.3. On the 1000 taxon datasets, ASTRID with a single iteration of FastME+NNI completed in between 1 and 3 seconds on every replicate, whereas ASTRID with a single iteration of FastME+SPR took between 18 and 43 seconds on the 20%, 50%, and 75% scaffold datasets; and between 7 and 15 seconds on the 100% scaffold datasets.

For the remainder of the analyses in this study, we will not show results run with more than one iteration of FastME, and we will only show the FastME-NNI and FastME-SPR variants of FastME.

5.4.1.3 Other distance methods

Earlier versions of ASTRID used BIONJ* as its distance method in the cases where the distance matrix was missing entries. Figure 5.1 shows that the new UPGMA protocol with FastME significantly reduces the error rate under the model conditions where BIONJ* returns inaccurate trees.

We also experimented with RapidNJ as the distance method for the second stage of the UPGMA protocol; its accuracy is better than UPGMA* alone or BIONJ*, but not as good as either FastME variant. It was, however, faster than FastME, taking an average of between 0.9 and 1.5 seconds on the 1000-taxon model conditions, while FastME with no local search took between 1.8 and 2.6 seconds.

Furthermore, Table 5.1 and Figure 5.3 show that the UPGMA protocol has a huge improvement in running time. For example, on the 75% scaffold 1000-taxon data, ASTRID-BIONJ* took an average of 1 hour 7 minutes, while ASTRID-FastME+NNI took just 2.5 seconds and ASTRID-FastME+SPR took 30 seconds. ASTRID-RapidNJ was slightly faster than ASTRID-FastME+NNI, and ASTRID-UPGMA was (by nature) faster than the rest.

Method	Running Time (HH:MM)	Memory usage (GB)	RF Error
ASTRID-RapidNJ	00:25	110	55%
ASTRID-FastME	04:22	132	26%
ASTRID-FastME+NNI	05:42	132	23%
MRL-FastTree	24:17	29	31%

Table 5.2: Running times and memory usage for 43,183 taxon simulated RNAsim-based dataset with 31 source trees. Source trees were simulated by sampling clade-based and scaffold trees from a 50,000 taxon RNAsim tree. FastRFS, ASTRAL, and MRL-RAxML were not able to run due to excessive memory usage. Calculations were run on a 20-core Intel Ivy Bridge cluster node with 256GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

5.4.2 Comparison to ASTRAL and supertree methods

ASTRID's accuracy is comparable to ASTRAL's, as seen in 5.2. On the 20% scaffold datasets, ASTRAL performed slightly better than ASTRID. However, ASTRAL is somewhat slower than ASTRID-FastME+SPR, and much slower than ASTRID-FastME+NNI, as shown in Table 5.1 and Figure 5.3.

MRL also has a similar accuracy to ASTRAL and ASTRID; it is, however, substantially slower than both of them, taking over 30 minutes to run on the 1000-taxon datasets that require 2 minutes for ASTRAL, 30 seconds for ASTRID-FastME+SPR, and under 3 seconds for ASTRID-FastME+NNI.

FastRFS is slightly more accurate than ASTRAL and ASTRID, but takes approximately as long as MRL to run, since the majority of its running time is taken by running MRL to enhance FastRFS's search space.

5.4.3 Scalability to very large datasets

On the large 43,183 taxon RNAsim dataset, ASTRAL failed to complete. It ran out of memory on a computer with 256GB RAM before even computing its constraint set. Since FastRFS requires ASTRAL's constraint set to run, it too was not able to run on this dataset.

ASTRID-RapidNJ was able to complete extremely quickly on this dataset, running in just 25 minutes with 110GB RAM. However, its error was quite high, at 55%.

ASTRID-FastME (no search) was slower, taking 4 hours 22 minutes and 132GB RAM. Its error was 26%.

ASTRID-FastME+NNI was slightly slower, at 5 hours 42 minutes, and also required 132GB RAM. Its error was lower, at 23%.

ASTRID-FastME+SPR also ran out of memory.

MRL with FastTree was able to run, and only required 29GB of RAM. However, it was much slower, taking 24 hours 17 minutes to complete, and had higher error than ASTRID-FastME+NNI, at 31%.

5.4.4 Biological dataset

On the 2,228 taxon CPL biological dataset with 39 source trees, we report running time and memory usage in Table 5.3. ASTRID-FastME+NNI was by far the fastest method, completing in just 13 seconds and requiring only 489MB RAM. ASTRAL was faster than ASTRID-FastME+SPR, but had much higher RAM usage (over 8GB) than any of the other methods. ASTRID-BIONJ* was much slower than the rest of the methods, requiring nearly six hours to run.

On the two smaller biological datasets, running times and memory usage are shown in Tables 5.5 and 5.4. On the 121-taxon seabird dataset, no method took more than a few seconds to complete. On the 558-taxon THPL dataset, FastRFS took 13 minutes, ASTRID-BIONJ* took just over 1 minute, arunnd ASTRAL and the iterative ASTRID variants completed in a few seconds.

On all these datasets, ASTRAL and ASTRID-BIONJ* had substantially higher memory usage requirements than the rest of the methods. This may be due to the fact that ASTRAL and BIONJ* are implemented in Java, while the rest of the methods are implemented in C or C++.

Method	Running Time (HH:MM:SS)	Memory usage (MB)
ASTRID-BIONJ*	5:47:00	8598
ASTRID-FastME+NNI	00:00:13	489
ASTRID-FastME+SPR	00:12:30	746
ASTRAL	00:01:02	5200
FastRFS	00:04:15	353

Table 5.3: Running times and memory usage for 2228-taxon comprehensive papilinoid legume (CPL) dataset with 39 trees [153]. FastRFS was run without MRL enhancement tree. Calculations were run on a 20-core Intel Ivy Bridge cluster node with 256GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

Method	Running Time (HH:MM:SS)	Memory usage (MB)
ASTRID-BIONJ*	00:00:04	80
ASTRID-FastME+NNI	00:00:00	6.9
ASTRID-FastME+SPR	00:00:00	7.4
ASTRAL	00:00:02	408
FastRFS	00:00:01	3.6

Table 5.4: Running times and memory usage for 121-taxon seabirds dataset [56]. FastRFS was run without MRL enhancement tree. Calculations were run on a 20-core Intel Ivy Bridge cluster node with 256GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

5.5 Discussion

Supertree estimation is an important tool for next generation phylogenomic analyses. The development of accurate supertree methods that can scale to datasets with tens or hundreds of thousands of taxa is

Method	Running Time (HH:MM:SS)	Memory usage (MB)
ASTRID-BIONJ*	00:01:12	1108
ASTRID-FastME+NNI	00:00:00	32
ASTRID-FastME+SPR	00:00:06	52
ASTRAL	00:00:06	2351
FastRFS	00:13:00	24

Table 5.5: Running times and memory usage for 558-taxon temperate herbaceous papilinoid legume (THPL) dataset [88]. FastRFS was run without MRL enhancement tree. Calculations were run on a 20-core Intel Ivy Bridge cluster node with 256GB RAM; FastME, ASTRAL and the ASTRAL subroutines in FastRFS are multithreaded.

critical for these projects. Existing state-of-the-art methods can scale to datasets with thousands of taxa, but struggle to go beyond that due to running time and memory utilization constraints.

The improved version of ASTRID presented here can perform analyses much larger than existing supertree methods. ASTRID's most significant advantage in the supertree context is its ability to scale to extremely large datasets while maintaining a low error rate. In particular, when the number of input trees is relatively low compared to the number of taxa, ASTRID's running time is dominated by the distance method used. This improved version of ASTRID allows fast distance methods to be used even in the supertree context when the ASTRID distance matrix is missing entries.

ASTRID-FastME+SPR is the most accurate version of ASTRID, and should be used if running time and memory constraints allow. However, the asymptotic runtime of the SPR step is $O(n^3)$, and the memory usage is empirically higher than with just NNIs, so scaling this technique beyond a few thousand taxa is unlikely.

ASTIRD-FastME+NNI is slightly less accurate than ASTRID-FastME+SPR under some conditions, but substantially faster and memory-efficient. It can scale to datasets with tens of thousands of taxa while maintaining low error rates.

For even larger datasets, ASTRID-RapidNJ may be able to scale beyond the level of FastME. However, it it is less accurate than FastME, so it is not usually the best method to use.

5.6 Conclusions

We have presented an improved version of ASTRID that significantly improves performance, in terms of accuracy and running time, in the supertree context. We have explored several ways to run ASTRID on a variety of datasets, and compared its performance to other leading methods, including ASTRAL, FastRFS, or MRL. ASTRID is competitive in terms of accuracy with these methods, and is able to scale to very large datasets that cannot be analyzed with any other method.

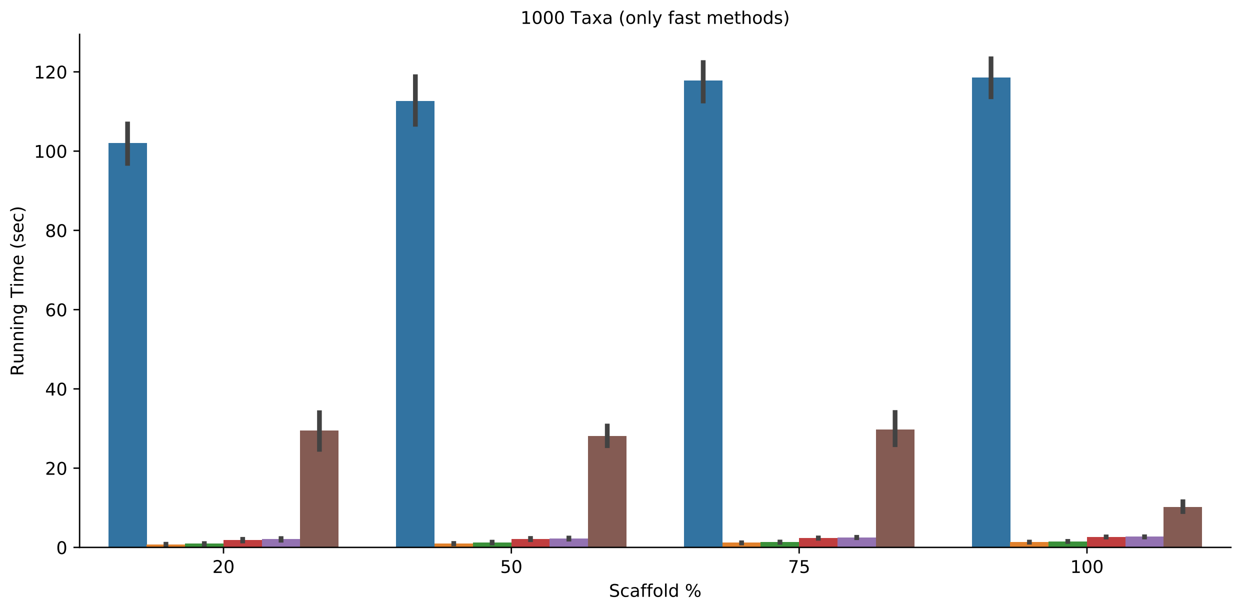


Figure 5.4: RF Error for 1000-taxon simulated datasets using up to 3 iterations of FastME with NNIs and FastME with SPRs. Each model condition has 10 replicates.

ASTRID allows for a variety of distance methods to be used, with slower but more accurate methods like FastME+SPR ideal for small to medium sized datasets, and faster methods like FastME+NNI and RapidNJ allowing for scaling to datasets with tens of thousands of taxa.

5.6.1 Future work

It is likely possible to further optimize ASTRID to limit memory consumption, particularly the built-in UPGMA* implementation. Some distance methods, including RapidNJ, are able to scale to extremely large datasets by intelligently storing data to disk when the entire distance matrix would not fit into memory. It may be possible to implement a similar approach in ASTRID, which would enable virtually limitless scaling.

Additional improvements to the distance methods used by ASTRID could also improve its performance. For example, it may be possible to directly add support for incomplete distance matrices to FastME. There may also be better methods than UPGMA* for the first stage of the ASTRID iterative protocol.

5.6.2 Additional figures

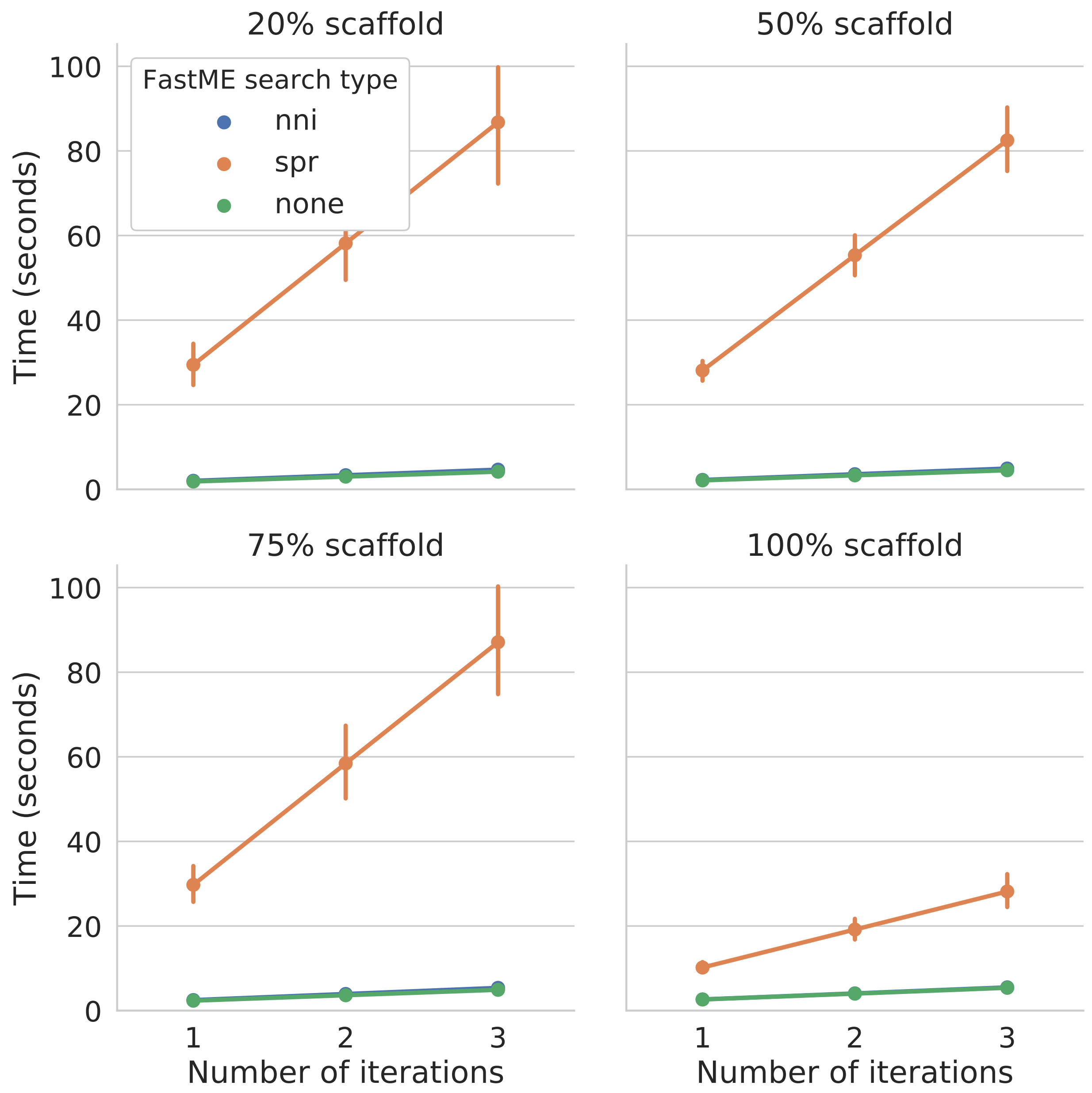


Figure 5.5: Running times for 1000-taxon simulated datasets using up to 3 iterations of FastME with NNIs and FastME with SPRs. Each model condition has 10 replicates.

Part III

Species Tree Estimation

Chapter 6

Species Tree Estimation with ASTRID

6.1 Background

Species tree estimation in the presence of gene tree incongruence is a major challenge for many biological analyses. Gene tree incongruence can result from a variety of processes, notably incomplete lineage sorting (ILS) [80], which is modelled by the multispecies coalescent (MSC) [57]. Concatenated maximum likelihood analyses is generally the most common method for species tree estimation from multiple loci, but can be statistically inconsistent, and even positively misleading, in some cases [116], thus converging to an incorrect tree with increasing amounts of sequence data.

In recent years, a number of species tree estimation methods have been developed that are statistically consistent under the MSC, and so will converge in probability to the true species trees as the amount of data increases; see [35, 59, 77]. Methods that are statistically consistent under the MSC include ASTRAL [95], ASTRAL-2 [94], *BEAST [47], BEST [73], the population tree from BUCKy [66], METAL [31], MP-EST [76], NJst [74], SNAPP [21], STEAC [78], STEM [61], and SVDquartets [25]. While little is yet known about some of these methods (either because they have not yet been adequately studied or because they are not yet implemented), only a few of them (MP-EST, NJst, and ASTRAL-2) have been shown to be able to analyze very large datasets (especially those with large numbers of taxa) with high accuracy. MP-EST has been used more than either NJst or ASTRAL-2, but NJst is more accurate than MP-EST, and ASTRAL-2 is more accurate than both [94]. Furthermore, the currently available implementation of NJst is slower than ASTRAL-2, and cannot run on some datasets [74, 94].

In this paper, we present ASTRID, a new ILS-aware distance-based method for species tree estimation. Our approach is based on NJst, but is substantially faster, and, unlike NJst, functions even when each gene tree contains only a small portion of the data. The input to NJst is a set of unrooted gene trees. In the first step, an $n \times n$ matrix $D[x, y]$ is computed, where $D[x, y]$ is the average distance (in terms of number of edges) between x and y among all the gene trees. In the second step, neighbor joining [120], a very popular distance-based method of phylogeny estimation, is used to produce the species tree.

ASTRID improves on NJst by enabling other distance-based methods to be used in the second step. In particular, although NJ cannot be run on datasets with missing entries, other distance-based methods can, and ASTRID enables the use of these other methods. We also explore the use of more accurate distance-based methods. Thus, ASTRID is a very simple modification to NJst. As we will show, ASTRID is much faster than NJst.

The comparison between ASTRID and ASTRAL-2 and MP-EST, two established coalescent-based summary methods, is also interesting. ASTRID completed in minutes on some datasets where the other methods took hours, and was fast enough to analyze datasets with 1000 species and 1000 genes on a single processor within an hour (ASTRAL-2 and MP-EST take much more time on datasets of this size). Furthermore, ASTRID clearly dominates MP-EST in terms of accuracy, and is competitive with ASTRAL-2 (more accurate in some cases, and less accurate in others). Finally, ASTRID has desirable theoretical properties: it runs in polynomial time, and it remains statistically consistent under the MSC model without assuming the molecular clock, nor requiring rooted gene trees as input.

6.2 Methods

6.2.1 ASTRID

The input to ASTRID is a set of unrooted gene trees $T_1, \ldots, T_k$. We let $S_i = \mathcal{L}(T_i)$ denote the leafset of T_i, and $S = \cup_i \mathcal{L}(T_i)$. Let $|S| = n$.

Step 1: Construct $n \times n$ matrix $\bar{M}$:

1. For all $i = 1, 2, \ldots, k$, compute $n \times n$ matrix M_i, as follows. For pairs p, q of species where both are in S_i, set $M_i(p, q)$ to be the number of edges in the path between p and q in T_i. For all other pairs p, q (i.e., where one or both are not in S_i), set $M_i(p, q) = 0$. Thus, the only non-zero entries in M_i are for pairs of species in T_i.
2. For all $\{p, q\} \subset S$, let $n(p, q)$ be the number of trees T_i that contain both p and q.
3. Define $n \times n$ matrix $\bar{M}$ by setting $\bar{M}(p, q) = \frac{\sum_i M_i(p,q)}{n(p,q)}$ if $n(p, q) > 0$, and $\bar{M}[p, q] = -1$ (to denote a missing value) otherwise.

Step 2: Compute tree on $\bar{M}$ using a selected distance-based method

6.2.2 Datasets

We tested species tree estimation methods on simulated datasets from previous publications, and also evaluated ASTRID on the mammalian biological dataset of 37 species, originally studied in [97]. Here we briefly describe the simulation procedures used to generate these datasets, and provide empirical statistics for the datasets in Table 6.1. See the original publications for details about the simulation protocols, and our supplementary online materials for links to the data.

All datasets included both true and estimated gene trees, obtained by using maximum likelihood methods on the true sequence alignments, as well as species trees estimated on these gene trees obtained in the prior publications. Each gene tree had at most one copy of each species. We computed ASTRID species trees for these datasets, using various techniques for Step 2 (how to compute the species tree given the distance matrix).

We estimated the amount of ILS in the data by quantifying the average gene tree discord in the data, using the average Robinson-Foulds (RF) [112] distance between true gene trees and the model species tree, expressed as a percentage (written AD for "average distance"). We also explored some simulated datasets where the DNA sequence evolution was under the strict molecular clock. Model conditions with AD at most 25% can be considered low ILS, conditions with AD between 26% and 39% can be considered moderate ILS, conditions with AD between 40% and 59% can be considered high ILS, and conditions with AD of at least 60% can be considered very high ILS. In Table 6.1, we indicate these ILS levels for the different model conditions we study both with the AD value, but also the general level (L for low, M for moderate, H for high, and VH for very high).

6.2.2.1 Mammalian and avian simulated datasets

These datasets were created in [98] to evaluate method performance under model conditions similar to real data. Species trees were generated with MP-EST for the avian phylogenomics dataset with 48 species and 14,446 loci [53], and for a mammalian dataset with 37 species and 447 loci [127]. These species trees were used as basic model trees, with branch lengths in coalescent units. In addition, two other model species trees were created for each dataset by scaling the species tree branch lengths up (to reduce ILS) or down (to increase ILS). The ILS levels of the resultant model species trees were very heterogeneous, ranging from AD = 21% (low) to 50% (high) for the mammalian simulation, and from AD = 29% (moderate) to 60% (very high) for the avian simulation.

Both datasets had sequences of length 500 for all three model conditions. For the default ("1X") branch length condition, the avian dataset also had sequences of length 250, 500, 100, and 1500, and the mammalian

Dataset	# genes	# taxa	ILS level (AD%)	# sites	ref.
Avian very high ILS (0.5X)	1000	48	60 (VH)	500	[98]
Avian high ILS (1X)	1000	48	47 (H)	250-1500	[98]
Avian moderate (2X)	1000	48	29 (M)	500	[98]
Mammalian high ILS (0.5X)	200	37	50 (H)	250-1000	[98]
Mammalian moderate ILS (1X)	200	37	29 (M)	250-1000	[98]
Mammalian low ILS (2X)	200	37	21 (L)	250-1000	[98]
10-taxon very high ILS	200	10	89(VH)	100	[13]
10-taxon high ILS	200	10	48 (H)	100	[13]
15-taxon clocklike	1000	15	82 (VH)	100-1000	[13]
ASTRAL-2 500K-1e6 (MC1)	1000	200	69 (VH)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC2)	1000	200	33 (M)	300-1500	[94]
ASTRAL-2 10M-1e6 (MC3)	1000	200	21 (L)	300-1500	[94]
ASTRAL-2 500K-1e7 (MC4)	1000	200	68 (VH)	300-1500	[94]
ASTRAL-2 2M-1e7 (MC5)	1000	200	34 (M)	300-1500	[94]
ASTRAL-2 10M-1e7 (MC6)	1000	200	9 (L)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC7)	1000	10	17 (L)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC8)	1000	50	30 (M)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC9)	1000	100	34 (M)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC10)	1000	500	34 (M)	300-1500	[94]
ASTRAL-2 2M-1e6 (MC11)	1000	1000	35 (M)	300-1500	[94]

Table 6.1: Empirical statistics of simulated datasets used in this study. The ILS level is measured by the average Robinson-Foulds distance (AD) between the true gene trees and the species tree, expressed as a percentage; ILS levels are then classified as low (L), moderate (M), high (H), or very high (VH).

dataset had sequences of length 250, 500 and 1000. Sequence evolution on these datasets deviated from the strict molecular clock.

6.2.2.2 10-taxon simulated datasets

These data were presented in [13], and explored two ILS levels (AD=48% (high) and AD=89% (very high)). Sequence evolution deviated from the strict molecular clock.

6.2.2.3 15-taxon clocklike simulated datasets

These datasets evolved under a strict molecular clock, and were presented in [13]. The species tree was a caterpillar model tree (i.e., a path with leaves hanging off the path) with very short internal branches, and a long branch to the outgroup species. The ILS level in these data was very high (AD=82%).

6.2.2.4 ASTRAL-2 simulated datasets

These data were presented in [94], and provided a variety of model conditions with varying ILS levels, tree shapes, numbers of taxa, and sequence lengths per locus. SimPhy [83] was used to generate the species and gene trees, based on two parameters: the number of generations (given as the first number in the model) and the speciation rate (given as the second number). The number of generations simulated ranged between 500K, 2M, and 10M, and the speciation rate varied between 1e6 and 1e7. Model conditions with fewer generations had more ILS. Model conditions with the 1e6 speciation rate had speciation events nearer the tips (leaves) of the trees, while model conditions with the 1e7 speciation rate had speciation events nearer the root. The ILS levels varied from very low (AD = 9%) to very high (AD = 69%). Sequences evolved down the gene trees under multiple GTRGAMMA models that deviated from the strict molecular clock. Maximum likelihood gene trees were computed using FastTree-2.

6.2.2.5 Incomplete gene tree datasets

To explore performance on incomplete gene trees, we modified the ASTRAL-2 dataset by randomly removing taxa from trees in the 50-taxon datasets. Up to 40 taxa were removed from the 50-taxon dataset, and up to 5 taxa were removed from the 10-taxon dataset. In each of these cases, maximum likelihood gene trees were estimated using FastTree-2 version 2.1.7 SSE3 [109], using the following command:

```
fasttree -nt -gtr -quiet -nopr -gamma -n 1000  <fastafile> > <genetreefile>
```

where `<fastafile>` was the input file of aligned sequences and `<genetreefile>` was the output file.

6.2.3 Distance-based tree estimation methods

In order to explore the design space for ASTRID, we ran various distance-based methods for Step 2 (computing the tree from the distance matrix). For incomplete distance matrices (where some entries are -1, indicating that the pair of taxa do not appear together in any gene tree), we explored the methods in PhyD* [28]: $NJ*$, $BIONJ*$, $MVR*$, $UNJ*$. These algorithms are all variants on neighbor joining that work on incomplete distance matrices. We also explored $FASTME$ [36], which is a heuristic for the minimum evolution problem.

6.2.4 ASTRAL-2

To compute ASTRAL-2 species trees on the incomplete gene trees generated for the ASTRAL-2 datasets, we ran ASTRAL-2 version 4.7.8, using command line arguments

```
java -Xmx4000M -jar astral.4.7.8.jar -i <genetrees> -o <outputtree>
```

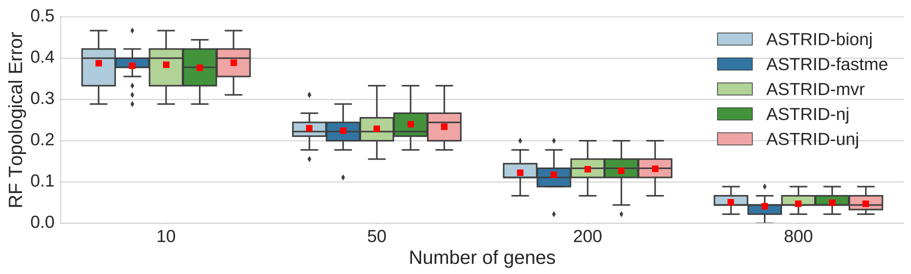


Figure 6.1: **A comparison of ASTRID variants on the moderate ILS avian simulated datasets with 500bp, using different distance-based methods for the tree estimation phase.** We report RF topological error rates over 20 replicates. Red dots represent means, while lines represent medians and boxes represent quartiles.

6.2.5 Computing tree error

All trees computed in this study were fully resolved. We report the RF tree error (the proportion of the branches in the model tree missing from the estimated tree), using scripts that are available in the supplementary online materials.

6.3 Results

6.3.1 Selection of distance-based tree estimation method for Step 2

First, we evaluated various distance-based tree estimation methods to determine which one would be most accurate for the tree computation phase of ASTRID. Results on datasets with all complete gene trees (no missing species in any gene) are shown in Figure 6.1 and results on datasets with incomplete gene trees are shown in Figure 6.2. Note that for datasets with entirely complete gene trees, FastME performed as well as or better than the other distance-based methods, but there were datasets with incomplete distance matrices in which FastME had very poor accuracy. Therefore, we selected FastME to analyze datasets where the distance matrix has no missing entries, since it had the best accuracy. For the datasets with incomplete distance matrices $\bar{M}$ (indicated by $\bar{M}[p,q] = -1$ for some p,q), we selected BioNJ*, since it generally had among the most accurate results of these PhyD* methods.

6.3.2 Comparison of ASTRID, ASTRAL, and MP-EST

We begin with a comparison between ASTRID, ASTRAL-2, and MP-EST on the avian simulated datasets with high (1X) ILS, varying number of genes and sequence alignment lengths, but where all genes are complete; see Figure 6.3.

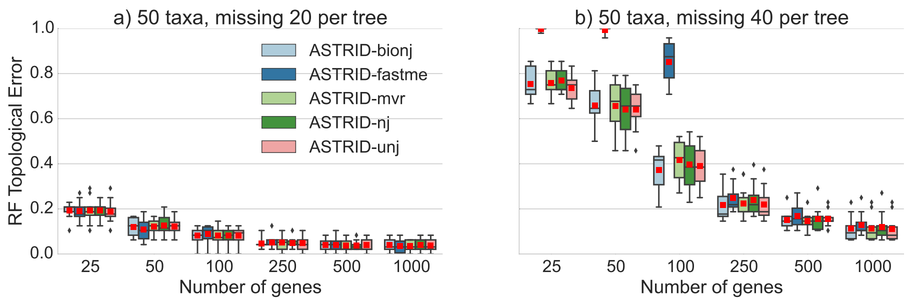


Figure 6.2: **Comparison of ASTRID variants on 50-taxon ASTRAL-2 MC8 datasets with missing taxa.** We show average RF error rates over 50 replicates for ASTRID variants, that differ in terms of the method used to compute the tree from the distance matrix. The datasets have taxa randomly removed from each gene and the sequence lengths truncated to 300bp. Red dots represent means, while lines represent medians and boxes represent quartiles.

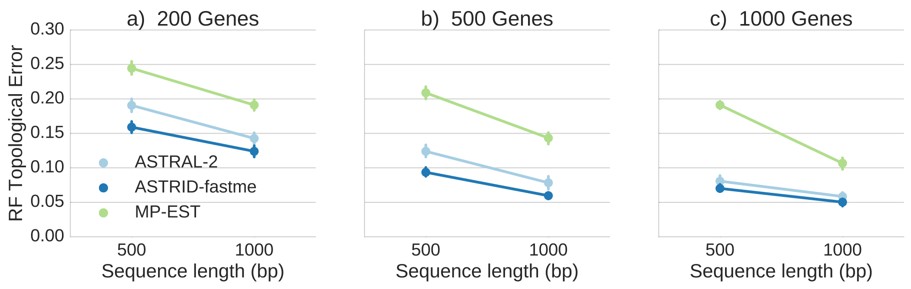


Figure 6.3: **Comparison of ASTRID, ASTRAL-2, and MP-EST on the avian simulated data.** The simulated data evolve under 1X (high ILS) species tree branch lengths, and with varying gene sequence lengths. We report mean RF rates with standard error bars over 20 replicates.

All methods improved with increasing numbers of genes or increasing sequence length; however, the methods differed substantially in terms of their accuracy. Across all conditions we explored, MP-EST had the highest error and ASTRID had the lowest error. ASTRAL-2 was in between, but was closer to ASTRID than to MP-EST. The gap between MP-EST and ASTRID was very large, and increased with the number of genes. For example, at 1000 genes and gene sequence alignments of length 500, MP-EST had 19% RF error while ASTRID had about 7% RF error. The gap between ASTRID and ASTRAL-2 was substantial on the 200- and 500-gene cases, but very small on the 1000-gene case.

Thus, although MP-EST is statistically consistent under the MSC model and hence theoretically robust to ILS, it did not have particularly good accuracy on these data. Among all coalescent-based methods, MP-EST is probably the one that has been used the most in biological data analyses, but its performance here and in [14, 94] demonstrates that it is not competitive with the best methods on datasets with even moderate numbers of species. Therefore, we omit MP-EST from the rest of this study.

6.3.3 Comparison of ASTRID and ASTRAL-2 on complete gene trees

Comparison on avian datasets. Figure 6.4 shows the performance of ASTRAL-2 and ASTRID on avian simulated datasets under three ILS conditions (moderate, high, and very high). Both methods performed better when provided with more genes, and both performed worse on higher levels of ILS. Overall, ASTRID tended to outperform ASTRAL-2, with the largest effect seen when many genes were available. With 800 genes available, the ASTRID species tree had a RF error rate that was 2.4 percentage points better than ASTRAL-2's under the very high and high ILS model conditions, and 1.2 percentage points better for the moderate ILS model condition. On the moderate ILS model condition, ASTRID had the greatest advantage over ASTRAL-2 for moderate numbers of genes. Above 200 genes, the error rate dropped below ten percent for both ASTRAL-2 and ASTRID, and ASTRID had an average advantage of only about one percentage point.

It is well known that summary methods improve in accuracy as the number of sites per gene or the number of genes increase [12, 42, 93, 117]. We explored the impact of varying the sequence length and number of genes on the avian datasets with high (1X) ILS, as well as on true gene trees. Figure 6.5 shows results on 10, 100, and 1000 genes; results on other numbers of genes have the same trends (data provided in supplementary materials). As expected, both methods improved with increased sequence length, and had their best accuracy on true gene trees. Both methods also improved as the number of genes increased. ASTRID was always at least as accurate as ASTRAL-2, with the biggest improvement for shortest sequences (with 250bp).

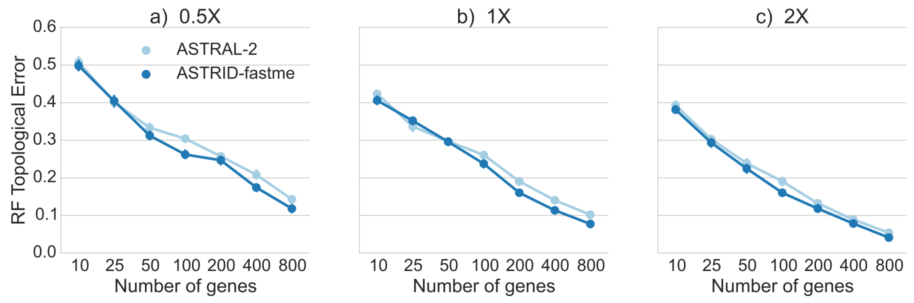


Figure 6.4: **Comparison of ASTRID and ASTRAL-2 on avian simulated datasets.** We show average RF error rates and standard error bars for 20 replicates. Gene sequence alignments have 500 sites and varying amount of ILS. Species tree branch lengths of 0.5x have very high ILS, 1X has high ILS, and 2x have moderate ILS.

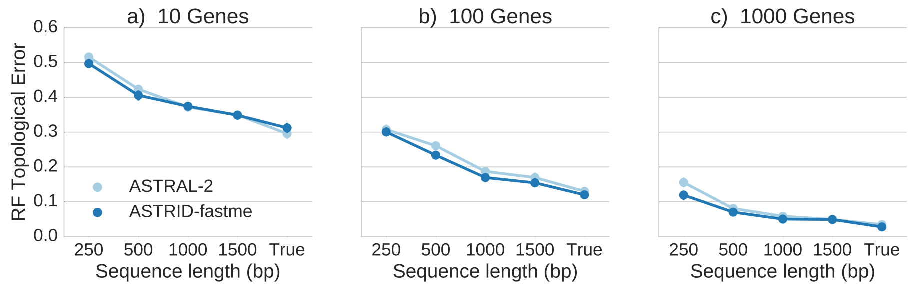


Figure 6.5: **Performance on the avian simulated data with 1X species tree branch lengths, varying gene sequence length and number of genes.** We report RF rates over 20 replicates.

Comparison on mammalian datasets. A comparison of ASTRAL-2 and ASTRID on the mammalian datasets with different levels of ILS (high, moderate, and low) is given in Figure 6.6. ASTRAL-2 and ASTRID performed fairly similarly on the low (2X branch lengths) and moderate (1X branch lengths) ILS conditions. Under the high ILS level (0.5X branch lengths), ASTRAL-2 was fairly consistently more accurate than ASTRID, with the largest improvement on the 10-gene case.

Comparison on the ASTRAL-2 datasets. We explored performance on the ASTRAL-2 datasets with 200 taxa (model conditions MC1 to MC6, see Fig. 6.7). These model trees varied in ILS level, with MC1 and MC4 having very high ILS, MC2 and MC5 having moderate ILS, and MC3 and MC6 having low ILS. Under MC2, MC3, and MC5, the two methods had essentially identical accuracy. However, under MC1, MC4, and MC6, ASTRAL-2 had an advantage over ASTRID. In MC1 and MC4, the improvement disappeared at 100 genes, but in MC6 ASTRAL-2 was still more accurate than ASTRID on 100 genes.

Comparison on the 15-taxon datasets. The 15-taxon datasets evolved on a caterpillar species tree under very high ILS (AD=82%), the highest ILS considered in this study. We explored performance under two sequence lengths (100bp and 1000bp) and varied the number of genes from 10 to 1000. Results on the

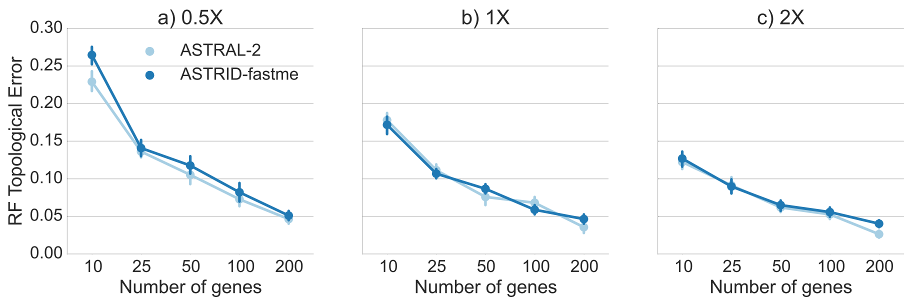


Figure 6.6: **Comparison of methods on mammalian simulated datasets, varying ILS level and number of genes.** We show average RF error rates and standard error bars for 20 replicates. Gene sequence alignments had 500 sites. Model conditions varied in ILS level from high (0.5x branch lengths) to low (2X branch lengths).

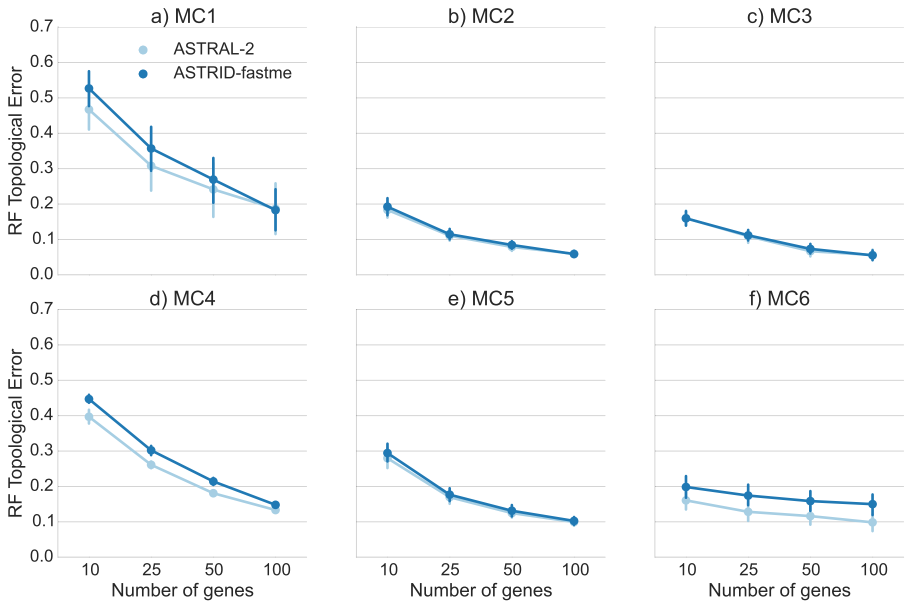


Figure 6.7: **Comparison of ASTRID and ASTRAL-2 on the simulated ASTRAL-2 datasets with 200 taxa, varying levels of ILS, tree shape, and number of genes.** We report RF error rates and standard error bars over 10 replicates. See Table 6.1 for information on the model conditions listed.

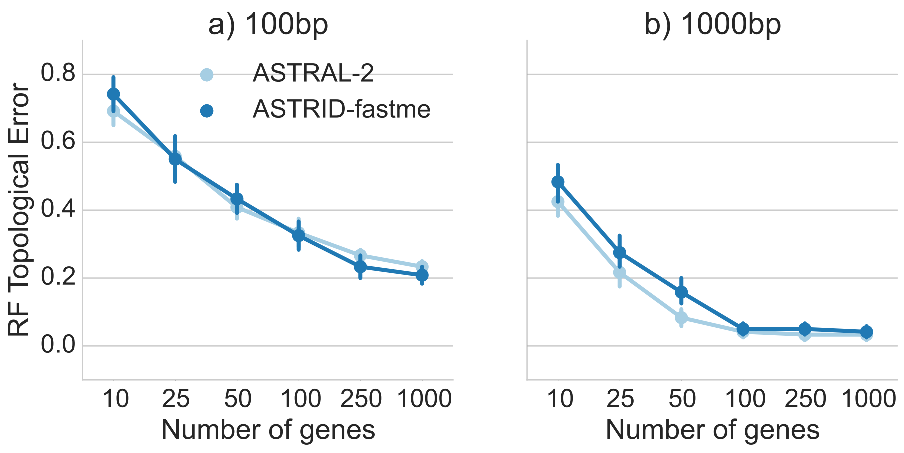


Figure 6.8: **A comparison of ASTRID and ASTRAL-2 on the 15-taxon simulated datasets for two different sequence lengths.** The 15-taxon datasets evolve down gene trees generated by a caterpillar tree with very high ILS (AD=82%), the highest ILS condition explored in this study. We report mean RF rates and standard error over 10 replicates.

15-taxon datasets (Fig. 6.8) showed very close performance between ASTRID and ASTRAL-2. On the 100bp alignments and on 1000bp alignments with at least 100 genes, the two methods could not be distinguished. However, on 1000bp alignments with at most 50 genes, ASTRAL-2 had an advantage over ASTRID.

Comparison on the 10-taxon datasets. The 10-taxon datasets evolved under two different ILS levels - high and very high, and we explored performance on both true and estimated gene trees; see Figure 6.9. In general, ASTRID and ASTRAL-2 had very close accuracy on these data, but there were some cases where they had different accuracy levels. For example, on the high ILS condition with estimated gene trees, ASTRAL-2 was more accurate than ASTRID for 200 genes, and ASTRID was more accurate than ASTRAL-2 on 25 genes.

6.3.4 Performance on incomplete gene trees

We explored the impact of missing data on ASTRAL-2 and ASTRID by deleting taxa from gene trees in the 50-taxon datasets (MC8) from the ASTRAL-2 collection, using 150bp per gene, and varying the number of genes and the amount of missing taxa; see Figure 6.10. ASTRAL-2 and ASTRID had very similar topological accuracy throughout these experiments. With low amounts of missing data (20% to 40% missing taxa from each gene tree), both methods had very good accuracy (below 5% tree error) by 500 genes. With 60% of the taxa missing from each gene tree, the error rates increased for low numbers of genes (above 20% RF error for up to 100 genes), but then declined to about 10% by 1000 genes. With 80% of the taxa missing from each gene (so that all gene trees have only 10 taxa out of 50), error rates were very high with 25 genes

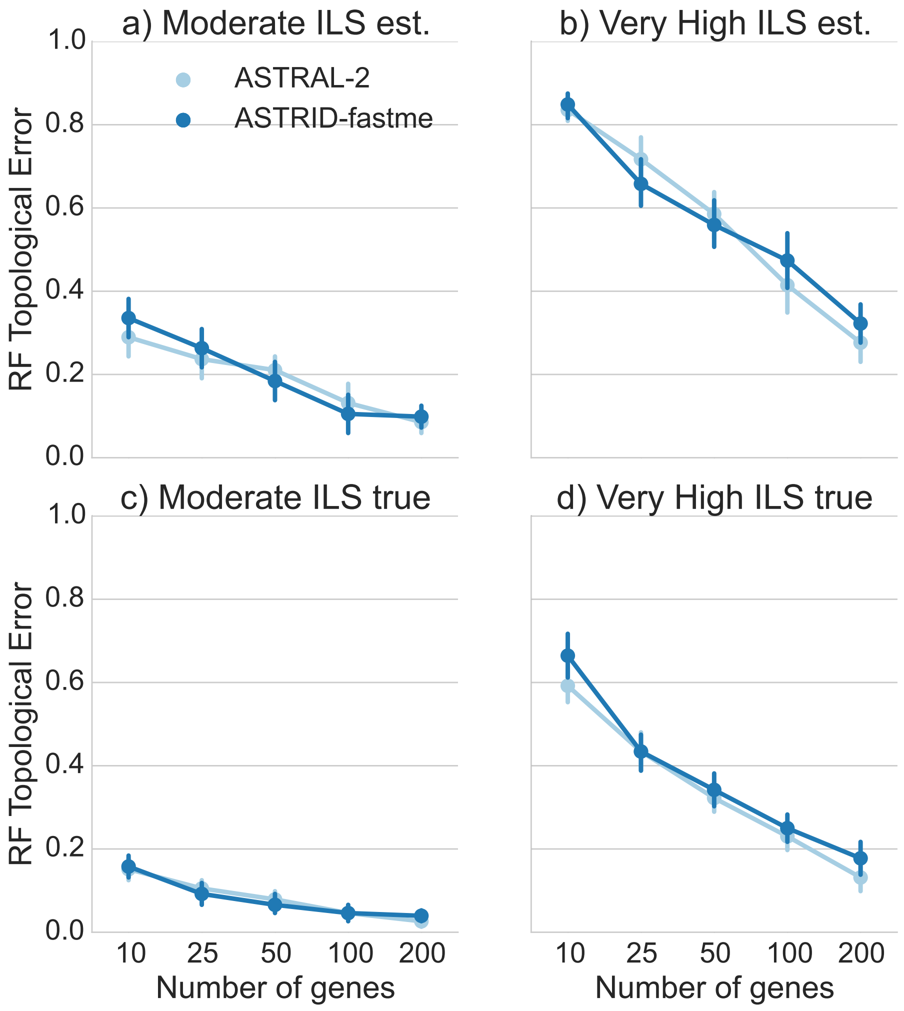


Figure 6.9: **Results on true and estimated gene trees on 10-taxon datasets with two ILS levels (high and very high).** All gene sequence alignments have 100bp. We report RF rates and standard error bars over 20 replicates.

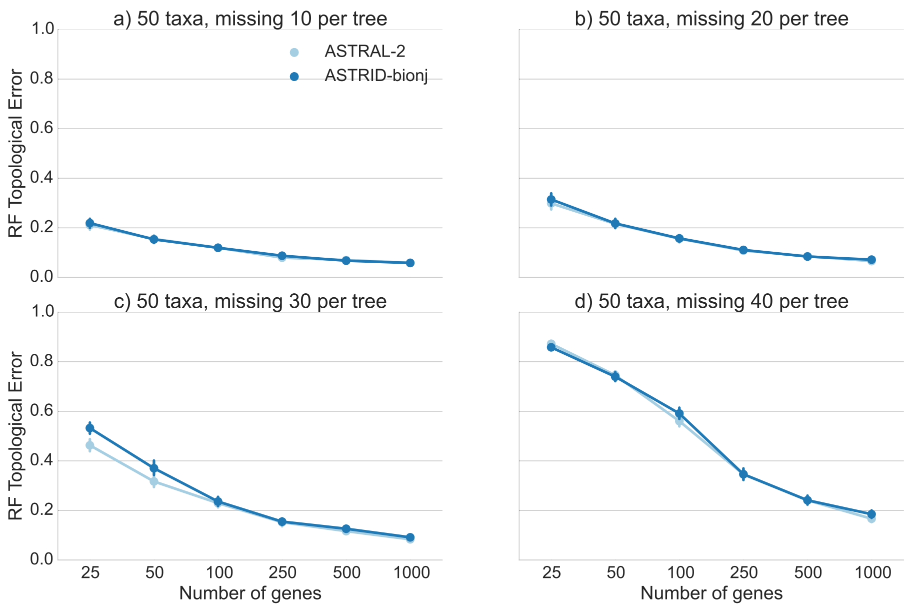


Figure 6.10: **Results on 50-taxon ASTRAL-2 dataset (MC8) with missing taxa and sequence lengths of 150bp.** We report RF rates and standard error over 50 replicates.

(at least 85% RF), but decreased quickly with increases in the number of genes, so that at 500 genes the error rate was 24%, and then at most 18% at 1000 genes. The trends suggest that the error rates had not plateaued, and that adding additional incomplete gene trees should result in continued improvement.

6.3.5 Analysis of the mammalian biological dataset

We analyzed the mammalian biological dataset originally studied in [127]. The original dataset had 37 species and 447 genes, but there were 23 erroneous genes (as noted by [97]) which we removed before doing the analysis.

We obtained maximum likelihood gene trees and bootstrap replicates of these gene trees from [97]. We then analyzed these data using ASTRAL-2 and ASTRID+FastME and compared these analyses to previously published trees obtained using ASTRAL and MP-EST [95]. We then annotated the branches of the ASTRID+FastME and ASTRAL-2 trees with bootstrap support from 100 multi-locus bootstrapping (MLBS). The ASTRID+FastME and ASTRAL-2 trees were topologically identical to the ASTRAL tree and differed only in the bootstrap support; see Figure 6.11 for the ASTRID+FastME tree. On the other hand, the support for the placement of Scandentia - one of the major open questions about mammalian evolution - was very low, only 47% (ASTRAL-2 gave it 82%). Hence, neither the ASTRID tree nor the ASTRAL-2 tree resolved the placement of Scandentia with high support.

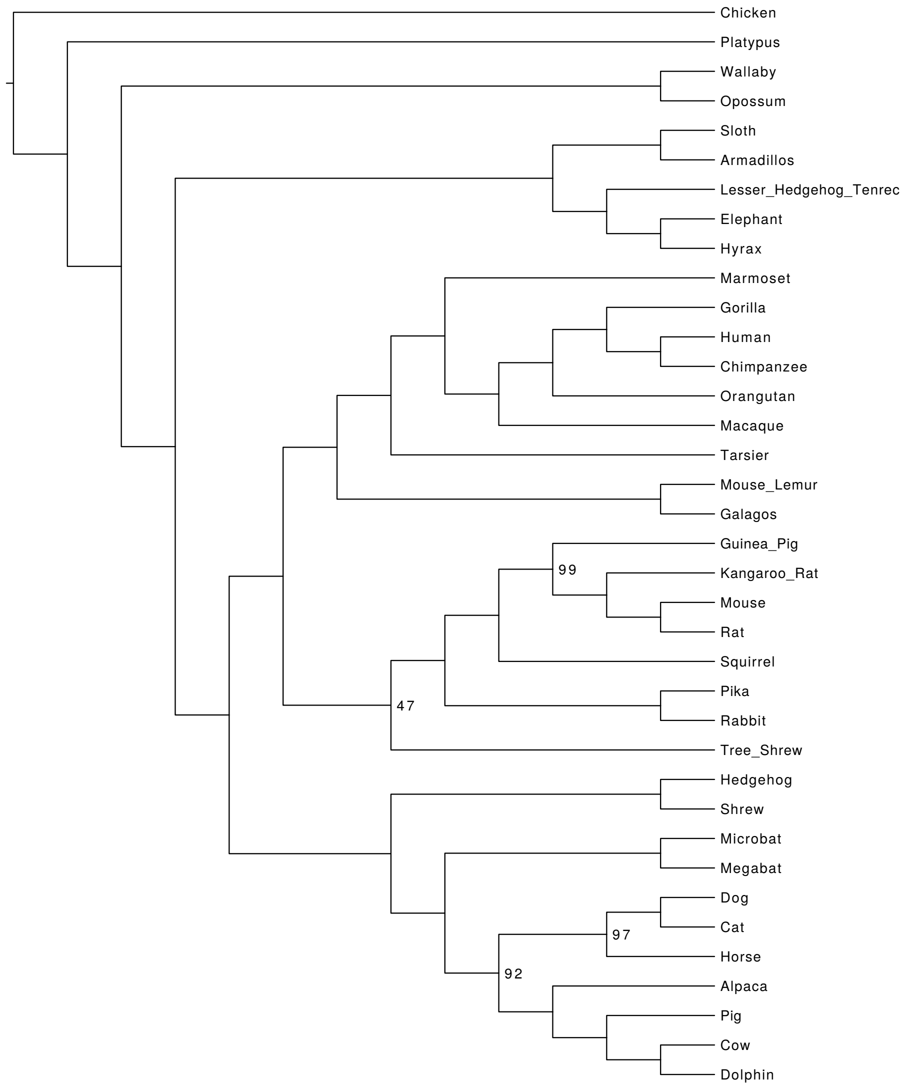


Figure 6.11: **ASTRID analysis of a mammalian biological dataset.** We used ASTRID+FastME to analyze the mammalian biological dataset studied in [95, 97], with 37 taxa and 424 genes. The branches are annotated with bootstrap support values from 100 MLBS bootstrap samples; values not shown indicate 100% support. The ASTRID tree is identical to the ASTRAL and ASTRAL-2 trees on the same data, but differs from the MP-EST analysis in the placement of Scandentia.

6.3.6 Running time results

6.3.7 Asymptotic running time

ASTRID has two steps: the first step computes the distance matrix, and the second step uses a selected distance-based method to construct a tree from the distance matrix. When the input has n species and k genes, then calculating the distance matrix can be performed in $O(kn^2)$ time. Distance-based tree estimation methods typically run in $O(n^2)$ to $O(n^3)$ time, but this step no longer depends on k. Hence, the overall running time depends on the selected distance-based method, but is generally dominated by the first phase, especially for typical inputs, for which $k >> n$. Thus, under the assumption that $k > n$ and that ASTRID uses a distance-based method that runs in $O(n^3)$ time, ASTRID's running time is $O(kn^2)$.

ASTRAL-2's scaling is more complicated to discuss. Asymptotically, ASTRAL-2 runs in $O(nk|X|^2)$ time, where n is the number of species, k is the number of genes, and X is a set of bipartitions it computes to constrain the search space. The size of X is not bounded by a polynomial in the input size, and the technique that ASTRAL-2 uses means that X can be large under conditions with high ILS. Thus the asymptotic running times of ASTRAL-2 and ASTRID (used with various distance methods) are quite different.

6.3.8 Running times on simulated data

In practice, creating the distance matrix took the majority of the running time. On 1000 taxa, creating the distance matrix took several minutes to several hours, depending on the number of genes, but running $FASTME$ took less than one second regardless of the number of genes. However, PhyD* methods were much slower than $FASTME$; on 1000 taxa, running any of the PhyD* methods took approximately 40 minutes (data not shown).

We recorded running times for ASTRAL-2, ASTRID-FastME, and NJst, on avian simulated datasets with high ILS (1X), as we varied the number of genes (see Fig. 6.12). Note that ASTRID-FastME was by far the fastest of the three methods, and NJst was the slowest. However, the trends suggest that NJst will be faster than ASTRAL-2 for larger numbers of genes. Note also that ASTRID-FastME and NJst both scaled linearly with the number of genes, but that ASTRAL-2's running time scaled super-linearly.

We recorded running times for two variants of ASTRID (one using FastME and the other using BioNJ*), and compared them to ASTRAL-2 on ASTRAL-2 simulated datasets with 1000 taxa (MC11) as we varied the number of genes (Fig. 6.13) and for 500-gene datasets in which we varied the number of taxa (MC 2 and 7-10, see Fig. 6.14). The relative running times show that all methods were very fast for smaller datasets, but were clearly distinguished on the larger datasets, where ASTRID-FastME was much faster than ASTRID-

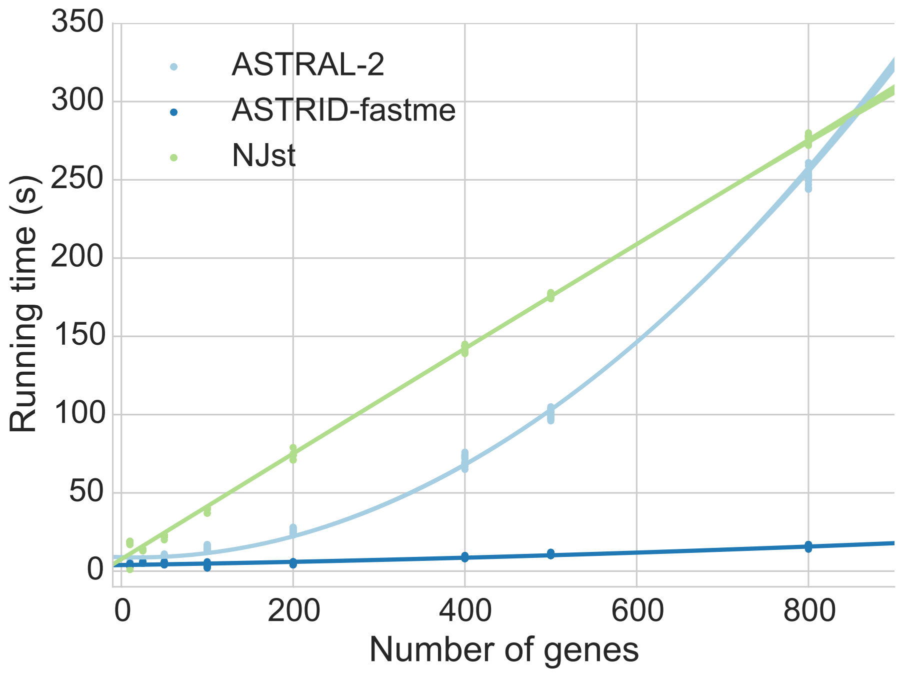


Figure 6.12: **Scatterplot of running times for ASTRID-FastME, ASTRAL-2, and NJst, on avian high ILS (1X) simulated datasets, varying number of genes.** We show running time for 20 replicates of each number of genes. The quadratic dependence of ASTRAL-2's running time is clearly contrasted with the linear dependence of both ASTRID and NJst. Experiments were run on a single core of a 2.7 GHz Intel Xeon processor.

BioNJ* and both variants of ASTRID were much faster than ASTRAL-2. For example, on the dataset with 1000 genes and 1000 taxa, ASTRID-FastME finished in 33 minutes, ASTRID-BioNJ finished in 1 hour and 10 minutes, and ASTRAL-2 finished in 12 hours and 30 minutes.

6.3.9 Running times on biological data

We recorded running times for ASTRID-FastME and ASTRAL-2 on the mammalian biological dataset. Both methods took 6 seconds for a single bootstrap replicate on one core of a 2.7 GHz Intel Xeon processor with 424 genes and 37 taxa.

6.4 Discussion

A few trends are apparent upon examining the data as a whole. ASTRAL-2 and ASTRID had, for the most part, very similar levels of accuracy, while MP-EST was consistently less accurate. However, there were cases where ASTRID and ASTRAL-2 have small but detectably different levels of accuracy. One intriguing trend in the data is the improvement of ASTRAL-2 over ASTRID on high ILS datasets; see Figures 6.6,

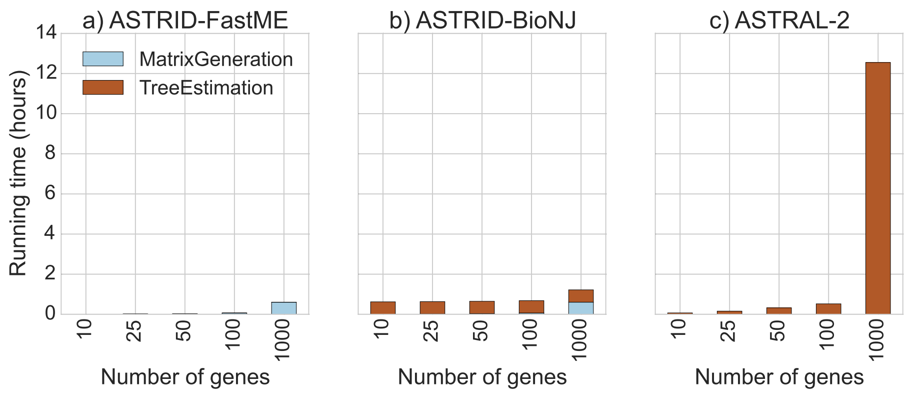


Figure 6.13: **Running time on the ASTRAL-2 simulated datasets with 1000 taxa (MC11), varying number of genes.** We show results for the each of the ASTRID steps – matrix generation and tree estimation. We compare ASTRID used with two ways of computing the trees: FastME and BioNJ*. Experiments were run on a single core of a 2.7 GHz Intel Xeon processor.

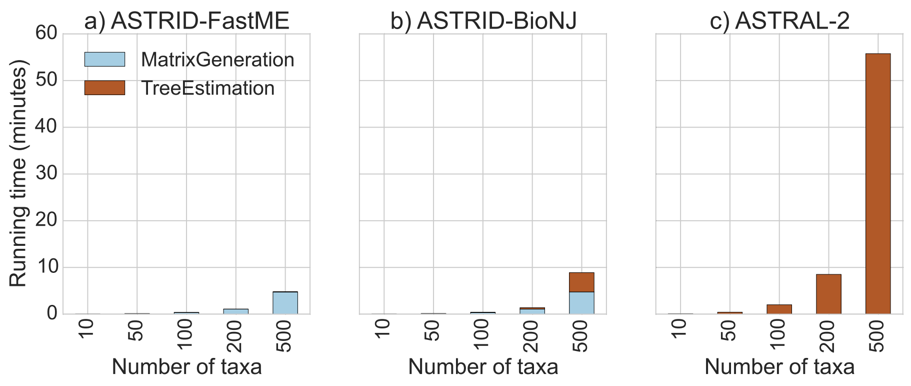


Figure 6.14: **Running time of two ASTRID methods on the ASTRAL-2 simulated dataset with 500 genes, varying number of taxa.** We show results on a single replicate of model conditions MC2 and 7-10 from the ASTRAL-2 collection. Experiments were run on a single core of a 2.7 GHz Intel Xeon processor.

6.7, 6.8, and 6.9. In particular, Figures 6.6 and 6.7 suggest that increases in ILS should favor ASTRAL-2 over ASTRID. Yet, ASTRID is consistently at least as accurate as ASTRAL-2 on the avian datasets, which have moderate to very high levels of ILS (Fig. 6.4). Thus, ILS level might have an impact on the relative accuracy of the two methods, but it is not a determining favor. Similarly, neither method dominates the other based on the number of taxa, number of genes, or amount of gene tree estimation error. Thus, it is very difficult to characterize the conditions under which each method is likely to have an advantage over the other. However, even for the cases where there are differences in accuracy, in general the differences are fairly small. Thus, the main difference between the two methods is computational efficiency, where ASTRID is clearly faster. ASTRID has the biggest running time advantage over ASTRAL-2 for large numbers of gene trees, since ASTRID scales linearly in the number of genes while ASTRAL scales super-linearly. This makes ASTRID an especially good method for genome-scale datasets that have a large number of genes.

6.5 Conclusion

ASTRID is a fast and highly accurate method for species tree estimation that is robust to high levels of ILS, and provably statistically consistent under the multi-species coalescent model. Like ASTRAL-2, ASTRID can analyze datasets with unrooted gene trees, an advantage that the two methods have over many other methods (e.g., MP-EST) that can only be run on rooted gene trees. ASTRID (like NJst) runs in time that is polynomial in the number of gene trees and species, but ASTRAL-2 and other leading coalescent-based methods do not have this guarantee. Thus, ASTRID has many desirable theoretical properties compared to existing methods.

From an empirical viewpoint, ASTRID is also extremely fast and can analyze very large datasets in minutes, where other methods either cannot run or take hours. In particular, ASTRID is much faster than ASTRAL-2, especially on datasets with many genes and large numbers of species. ASTRID also produces more accurate trees than MP-EST and NJst, and is competitive with ASTRAL-2 in terms of accuracy.

However, even better (more accurate) results might be obtained through more extensive modifications to the ASTRID algorithm design. In particular, the accuracy of the tree depends on the particular distance-based method that is used. New distance-based phylogeny estimation methods, such as the absolute fast converging methods [113, 145], might provide improved accuracy for very large datasets with many thousands of species. Another important direction is developing additional methods for estimating species trees from distance matrices that have good accuracy when the distance matrix has missing data. As we saw here, FastME produced more accurate trees than the PhyD* methods, but it could only be applied to distance matrices without any missing data. An extension of FastME to enable it to handle incomplete distance matrices would also be of great interest.

This study can be expanded in several directions. Future work should more carefully investigate the conditions under which ASTRID is more reliable than ASTRAL-2, and explore performance on more biological datasets. This study also only investigated relatively long sequences; a subsequent study should investigate the relative and absolute accuracy of ASTRID and other methods on very short sequences, since recombination-free loci can be very short [42]. In addition, this study only examined datasets with a single individual per species, yet ASTRID (like NJst) can be run on datasets with multiple individuals; future work should evaluate the absolute and relative accuracy of ASTRID and other methods on such data. This study showed that ASTRID performed well in terms of species tree topology estimation, but we did not explore its accuracy with respect to the estimation of coalescent branch lengths; future work will need to explore how well ASTRID estimates these numeric parameters. Finally, it may well be that ASTRID will be most useful as a starting tree for use within more computationally intensive analyses, including Bayesian MCMC analyses (e.g., *BEAST) or maximum likelihood analyses.

6.6 Additional Implementation details

6.6.1 Fast distance matrix calculation

6.6.2 Distance methods

6.6.3 Handling multiple individuals per taxon

Chapter 7

SVDquest

7.1 Introduction

Species tree estimation is a fundamental part of many biological analyses. Evolutionary processes such as incomplete lineage sorting [80], gene duplication and loss [107], and horizontal gene transfer [151] can result in heterogeneity across the genome, so that different parts of the genome have different trees. Much of the recent literature has focused on species tree estimation in the presence of incomplete lineage sorting (ILS), as - according to the multispecies coalescent model (MSC), which models gene tree heterogeneity due to ILS - all large-scale phylogenomic studies are expected to be affected by ILS to some extent. Furthermore, recent research has established that one of the most commonly used methods for species tree estimation, unpartitioned concatenation using maximum likelihood (CA-ML), can be statistically inconsistent under the MSC, and may converge to the wrong tree with probability converging to 1 as the number of loci increases for some model species trees [115]! Simulation studies evaluating CA-ML have also shown model conditions where it can produce highly supported incorrect trees in the presence of sufficiently high ILS, as well as other model conditions in which it performs well and may possibly be statistically consistent (e.g., [34, 62]).

New approaches for constructing species trees that are statistically consistent under the MSC have been developed (see [6, 82] for recent reviews) and used in a number of biological analyses (e.g., [49, 53, 99, 127, 150]). Many of these approaches operate by computing gene trees on different loci and then combine these estimated gene trees into a species tree, and some of these "summary methods" (e.g., MP-EST [75], NJst [74], ASTRAL [92, 94, 156], and ASTRID [141]) can give highly accurate results in practice in circumstances where CA-ML performs poorly due to high levels of gene tree discordance [75, 93, 141]. Furthermore, some summary methods are typically very fast and can analyze large datasets. As a result, summary methods have become standard approaches for estimating species trees when ILS is suspected.

However, the proofs of statistical consistency for summary methods depend on having accurate gene trees [117], which is generally not expected on biological datasets. In addition, the proofs depend on all sites within each locus evolving down a single tree (i.e., c-genes), and meeting that requirement can result in very short sequences for each locus [128], which increases gene tree estimation error. Furthermore, from an empirical standpoint, there is ample evidence that gene tree estimation error increases the error of species trees estimated using summary methods [12, 33, 42, 51, 89, 93, 100, 108, 128], and that CA-ML can be more accurate than even the most accurate summary methods when gene tree estimation error is sufficiently high, even in the anomaly zone (see [100] and references therein).

The impact of gene tree estimation error on species tree estimation has led to interest in methods that can estimate species trees without needing to compute gene trees, and that are statistically consistent under the MSC. One such approach is to co-estimate gene trees and species trees; *BEAST [47] and BEST [72] are two such methods, but both are very computationally intensive [12, 58, 68, 86, 159]. Another type of approach estimates the tree directly from the observed site pattern frequencies using properties of the MSC, and does not also try to estimate gene trees; examples of such methods include SuperMatrix Rooted Triple (SMRT) [34], SNAPP [21], SVDquartets [25], and METAL [31, 101]. PoMo [32] and its improved version revPoMo [122] can also be considered in this category, although these methods are not established to be statistically consistent under the MSC. These "site-based" methods are considered particularly suitable for datasets generated using phylogenomic protocols such as RADseq that produce loci with very few variable sites, which makes highly accurate gene tree estimation unlikely [106].

The most popular of these site-based methods is available in PAUP* [136] and operates as follows. Given a multi-locus dataset, the loci are concatenated into a single long alignment. Then, for each set of four species, a quartet tree for that set is computed using SVDquartets. Finally, a species tree is sought that agrees with as many of these quartet trees as possible. The number of quartet trees that the species trees satisfies is called its MQSST score, where MQSST refers to the *Maximum Quartet Support Species Tree*, and the problem of finding the tree with the highest MQSST score is the MQSST problem. Because the MQSST problem is NP-hard [54], PAUP* uses a heuristic search to seek a good solution to MQSST. This method, which we refer to as SVDquartets+PAUP*, is increasingly popular in phylogenomics studies [4, 8, 18, 22, 29, 46, 48, 49, 67, 68, 84, 85, 99, 103, 106, 148, 149].

We present SVDquest, a new site-based method for estimating species trees in the presence of ILS. SVDquest has the same basic approach as SVDquartets+PAUP* in that it uses SVDquartets to estimate quartet trees, and then combines these quartet trees into a species tree; the difference between SVDquest and SVDquartets+PAUP* is the technique each uses to combine the quartet trees. Instead of employing a heuristic search strategy, SVDquest uses dynamic programming (an algorithm design technique) to find a provably optimal solution to the MQSST problem within a constrained search space. The constraints are defined by a set of *allowed bipartitions on the species set*, and we use the dynamic programming algorithm from [20] to find a species tree that maximizes the quartet support score within that constrained search space. If the search space is not constrained, then SVDquest finds a globally optimal solution to MQSST but will run in time that is exponential in the number of species, and so be too computationally intensive to use on datasets with more than about 15-20 species. However, we show that we can constrain the search space so that the algorithm runs in polynomial time and finds very good solutions to its optimization problem. Furthermore, by selecting the bipartitions appropriately, the new method, which we refer to as SVDquest*, is *guaranteed* to satisfy at least as many quartet trees computed by SVDquartets as SVDquartets+PAUP*.

We present results from an extensive performance study using both simulated and biological datasets. We find that SVDquest* finds better MQSST scores than SVDquartets+PAUP* under most conditions, particularly under higher levels of ILS and gene tree estimation error. We compare SVDquest* to a set of leading species tree estimation methods. We include two summary methods ASTRAL [92, 94] and ASTRID [141], because these two methods have been shown to have high accuracy under a wide range of model conditions and are both statistically consistent under the MSC (again, under the assumption that each is given true gene trees). We also include CA-ML (using RAxML), since (as noted earlier) trees computed using CA-ML are often at least as accurate as trees computed using summary methods. We do not include any of the co-estimation methods, as they are too computationally intensive to use on the datasets we explore.

The relative performance between these methods depends on the model condition (and in particular on the amount of gene tree estimation error), but SVDquest* has dramatic improvements over the summary methods under conditions with high average gene tree estimation error and many genes. CA-ML has the best accuracy of all methods under conditions with low to moderate ILS, but the coalescent-based methods outperform CA-ML when ILS is sufficiently high.

Finally, we also show that returning the strict consensus of the optimal trees computed by SVDquest* provides further improvements in topological accuracy. Thus, SVDquest* is a new method with improved accuracy compared to existing coalescent-based species tree methods under a range of realistic model conditions.

7.2 Materials & Methods

7.2.1 SVDquest

The input to SVDquest is a set of sequence alignments (one alignment for each locus). Phase 1 of SVDquest uses the SVDquartets implementation in PAUP* to compute a set of quartet trees and Phase 2 combines these quartet trees into a species tree on the full set of species. Thus, SVDquest is identical to SVDquartets+PAUP* in Phase 1, but differs from SVDquartets+PAUP* in Phase 2. Specifically, SVDquest uses dynamic programming to find an optimal solution to the MQSST problem within a constrained search space, similar to how ASTRAL and FastRFS [144] (a method for the Robinson-Foulds Supertree problem) find optimal trees for their optimization problems within constrained search spaces. Furthermore, since there can be more than one optimal tree for the MQSST problem, we also consider an optional Phase 3, in which a consensus tree is computed on the optimal trees found in Phase 2 using SIESTA [142], a method that can be used with dynamic programming methods (such as SVDquest) that solve constrained optimization problems. As shown in [142], the use of SIESTA with ASTRAL and FastRFS to compute the strict consensus tree typically results in an improvement in overall topological accuracy, suggesting that SIESTA might also improve SVDquest*.

Phase 1: computing the set Q of quartet trees. Phase 1 (the quartet tree estimation phase) computes an unrooted binary tree for every set of four species. For each set of four species, we use SVDquartets as implemented in PAUP* to select the best of the three possible quartet trees, and we refer to the set of all quartet trees computed in this way by Q. In some cases, SVDquartets will not return a tree on a set of four species because the scores are too close (e.g., this can happen when there are too few variable sites).

Phase 2: computing an optimal species tree from using Q. Phase 2 (the species tree estimation phase) uses the quartet trees computed in Phase 1, and attempts to find an optimal solution to the MQSST problem. Since MQSST is NP-hard, algorithms for finding the globally optimal solution are not scalable. Hence, SVDquest typically operates in a mode where instead of searching for a globally optimal tree, it finds an optimal tree within a constrained search space.

SVDquest has three modes: *unconstrained, constrained-basic*, and *constrained-enhanced.* The *unconstrained* mode does not constrain the search space at all and hence is the most computationally intensive; it can only be used when the number of species is small enough (up to 15-20 taxa). In the two constrained modes, the search space is defined by a set X of bipartitions on the species set, and the constraint is that the output species tree *must* draw its bipartitions from X. Therefore, if X is all possible bipartitions on the species set then there is no constraint on the set of species trees that can be returned; otherwise, there is a reduction in the set of species trees that can be considered during the search for the best tree.

To compute the basic set X of allowed bipartitions, SVDquest uses the following protocol. First, it computes maximum likelihood trees on every gene; then, it runs a subroutine in ASTRAL-II [94] to compute a set X of bipartitions that is guaranteed to include all the bipartitions from the input gene trees. The enhanced set of allowed bipartitions is computed by adding bipartitions to X. For example, we can add the bipartitions in the SVDquartets+PAUP* tree to X; we refer to this *constrained-enhanced* variant as SVDquest*. Since SVDquest exactly solves the MQSST optimization problem within the constrained search space, the SVDquest* tree is *guaranteed* to have a MQSST score that is at least as good as the SVDquartets+PAUP* tree's MQSST score.

Phase 3: Returning the strict consensus of the set of optimal trees computed in Phase 2. As noted, there can be more than one species tree that has an optimal MQSST score within the constrained space. Hence we provide an optional Phase 3 in which we use SIESTA to compute the strict consensus of all the trees in that set.

SVDquest and SVDquartets+PAUP* are techniques that compute a set Q of quartet trees using SVDquartets and then attempt to find a species tree that satisfies the maximum number of quartet trees in Q. The key difference between these methods is how each it solves the MQSST problem. SVDquest and its variants use a polynomial time dynamic programming algorithm from [20] to provably solve MQSST within a constrained search space; in contrast, PAUP* uses other techniques to attempt to solve MQSST that do not provide guarantees of optimality within any constrained search space, but have the benefit of not being explicitly constrained to a subset of the search space.

7.2.2 Datasets

We explored performance on 10-, 15-, and 50-taxon simulated datasets. We also analyzed a mammalian biological dataset with 37 species that was first studied in [127], and later used to compare coalescent-based methods [12, 92, 128]. The mammalian dataset originally included 447 loci, but 21 of these had mislabeled sequences and two were clear outliers [92], so we excluded them from our analysis. This left 424 loci and a total of 1,338,678 sites in the concatenated alignment. We obtained the alignments and RAxML gene trees from [127], and we obtained the CA-ML tree on the 424 loci from [92]. The average bootstrap support on the gene trees was 71%. The main questions are the positions of two groups: *Chiroptera* and *Scandentia*.

The simulated datasets were derived from prior publications, described individually below. Each dataset has model gene trees that evolve down model species trees under the multi-species coalescent, and indel-free sequence alignments evolved down those gene trees under standard site evolution models. Each gene is a proper c-gene (i.e., there is no recombination within any gene), and unless specified otherwise, the strict molecular clock assumption is not enforced. In some cases, we combined sequence data from different genes together (to simulate failure to detect recombination) by concatenating sequences simulated on gene trees with different topologies; this produces a set of "supergene" datasets.

Statistics for the simulated c-gene datasets are presented in Table 7.1. To characterize the level of ILS, we use the AD value, which is the average normalized Robinson-Foulds (RF) distance [112] between model gene trees and model species trees (i.e., the percentage of the non-trivial bipartitions in the true gene tree that do not appear in the true species tree). We also report gene tree estimation error (GTEE), which is the average normalized RF distance between model gene trees and estimated gene trees.

The 50-taxon datasets were simulated with SimPhy [83] and were originally presented in [94]. These datasets have 50 taxa, 1000 loci, and 300-1500 sites per locus. The original versions of these datasets have 200 taxa, but 150 taxa have been randomly removed from each replicate to reduce the time and memory requirements of the analysis. This dataset contains three model conditions with three different ILS levels of 13%, 33%, and 72% AD. Loci were originally of variable length, but we reduced the sequence lengths for these experiments to 25, 50, 100, and 300 sites. These datasets have a speciation rate of 10^{-6}, resulting in speciation close to the tips of the model trees (i.e., recent divergence). Sequences evolved with a GTR+Gamma model and no molecular clock.

There were 26 model conditions (all with very high ILS) on which SVDquartets+PAUP* and SVDquest failed to return a tree. In these cases, PAUP* reported that there were "No informative quartets found", and examining the sequences showed that there were very parsimony-informative sites. Hence, we report results only on those replicates for which all methods completed, a number that varies from 48 to 50 for each model condition. See Supplementary Materials, Section 2 for additional details.

The 15-taxon simulated datasets (from [13]) have very high ILS (82% AD), and have 1000 loci with 1000 sites evolved with a strict molecular clock using a GTR+Gamma model (i.e., the gene trees are ultrametric). Model species trees all have the same "caterpillar" topology, and gene trees obey a strict molecular clock. We used 10, 100, and 1000 sites per locus, with an average of 65%, 53%, and 18% GTEE, respectively. Each model condition has 10 replicates, and all of them completed successfully with all methods.

The 10-taxon simulated datasets (also from [13]) have two ILS levels (43% and 84% AD). These datasets have 200 loci with 10, 50 and 100 sites per locus, and GTEE levels between 40% and 75%. Species trees were randomly generated under a Yule process, gene trees are not ultrametric, and sequence data evolved under a GTR+Gamma model. Each model condition has 20 replicates; however, as with the 50-taxon datasets, some replicates had too few parsimony-informative sites, so that SVDquartets failed to compute any quartet trees. This occurred for 10 replicates of each model condition (combination of ILS level, number of genes, gene sequence length), and we report results for the other 10 replicates of each model condition.

Number of taxa	50	15	10
Number of loci	50, 100, 500, 1000	50, 100, 1000	25, 50, 200
Locus length	25, 50, 100, 300	10, 100, 300	10, 50, 100
GTEE	15%-100%	15%-72%	40%-75%
AD% (ILS)	13%, 33%, 72%	82%	43%, 84%
Number of replicates	40-50	10	10
Strict molecular clock?	No	Yes	No

Table 7.1: Summary of simulated datasets. GTEE is gene tree estimation error (i.e., the average normalized Robinson-Foulds distance between the estimated and model gene trees). AD% measures the average normalized Robinson-Foulds distance between the model gene trees and the model species tree, and is due only to ILS.

7.2.3 Species tree methods for comparison

We compared SVDquest* to ASTRAL v4.10.2 [94], ASTRID v1.4 [141], SVDquartets+PAUP* as implemented in PAUP* v4.0a151 [136], and unpartitioned concatenated maximum likelihood (CA-ML) under a GTR-GAMMA model using RAxML v8.2.6 [130]. The same version of RAxML was used to estimate trees on gene sequence alignments and supergene sequence alignments under a GTR-GAMMA model. We ran the Windows version of PAUP* using WINE v1.6.2 due to its improved numerical routines compared to the Linux version. Exact commands for all methods are supplied in the appendix.

7.2.4 Evaluation criteria

On the simulated datasets we use Dendropy [132] to evaluate estimated trees for topological accuracy. All model species trees are binary (i.e., fully resolved), but some of the estimated species trees are not binary; hence, we report the average of the false positive and false negative rates; this is identical to normalized Robinson-Foulds (RF) error rates [112] when the estimated trees are binary.

On the mammalian biological dataset, we evaluated the estimated species trees using established clades, taking branch support into account. For branch support on trees computed using the summary methods and CA-ML, we used the local posterior probability branch support technique [121] in ASTRAL, which is based on the initial set of estimated gene trees and has been shown to produce better estimates of the probability of a branch being accurate than multi-locus bootstrapping [121]. Branch support of species trees computed using site-based methods such as SVDquartets+PAUP* or CA-ML is commonly performed using non-parametric bootstrapping, but this approach is computationally intensive because it requires the calculation of species trees for all the bootstrap replicates. For this reason, we used a modified non-parametric bootstrap support technique described below (with 100 bootstrap replicates) to produce estimates of the branch support for the SVDquest* tree, and we compare the branch support we obtained using this modified non-parametric bootstrapping technique to the support we receive using the usual non-parametric bootstrapping technique. The modification to non-parametric bootstrapping that we use is very simple, and provides an approximation to the branch support that would be obtained using full non-parametric bootstrapping. We compute the constraint set X of bipartitions using the original dataset (i.e., not bootstrapped). Then, for each of the 100 bootstrap replicates, we run SVDquartets to compute the quartet trees, and we run SVDquest on the quartet trees computed by SVDquartets, using the constraint set X. In every other respect, the estimation of branch support we use follows the same protocol as with the usual non-parametric boostrapping procedure. Note that this approach has the benefit that for every bootstrap replicate the quartet trees are based correctly on SVDquartets, and only differs from full non-parametric bootstrapping in how the search space is constrained. Hence, this branch support technique does not affect the MQSST score of any returned tree, and only constrains which trees are considered permitted solutions.

7.2.5 Experiments

Experiment 1: Comparing SVDquest* and SVDquartets+PAUP* on simulated c-genes with respect to MQSST scores. The goal of this experiment is to determine whether SVDquest* finds better MQSST scores than SVDquartets+PAUP*. We tested both methods on simulated and biological datasets and reported MQSST criterion scores.

Experiment 2: Comparing coalescent-based species tree estimation methods on simulated c-genes with respect to tree topology. In the second experiment, we evaluated SVDquest*, SVDquartets+PAUP*, ASTRAL, and ASTRID, with respect to tree topology accuracy on a wide range of simulated datasets where all genes are c-genes (i.e., for each gene, all the sites evolve down a common tree topology).

Experiment 3: Comparison of coalescent-based species tree methods on multi-locus supergene datasets. In this experiment, we explored SVDquest*, SVDquartets+PAUP*, ASTRAL, and ASTRID on multi-locus datasets where the c-genes are randomly combined into supergenes (with the same number of c-genes), so that the assumption that all the sites in a given locus evolve down the same tree is violated. This experiment is motivated by the real-world challenge of failing to detect recombination events within gene sequence alignments. We estimated ML trees on these supergene alignments, and then used these "supergene trees" as the input for ASTRAL and ASTRID. Since SVDquartets computes quartet trees using all the sites in the concatenated alignment, this does not impact SVDquartets; it also does not change MQSST scores for any estimated species tree, as these are based on the quartet trees computed using SVDquartets. Hence, the use of supergenes does not impact SVDquartets+PAUP*. However, the use of supergenes instead of genes impacts summary methods, since the supergene trees will not be equal to the gene trees. It also affects SVDquest and SVDquest*, since it can change the constraint space that it computes using ASTRAL.

Experiment 4: Comparison of coalescent-based methods to CA-ML on simulated datasets. In this experiment, we compared CA-ML to SVDquest*, ASTRAL, and ASTRID on all simulated datasets with respect to the normalized Robinson-Foulds topological error rates.

Experiment 5: Comparison of coalescent-based methods on a mammalian biological dataset. We compare the SVDquest* tree on the mammalian biological dataset to trees computed using SVDquartets+PAUP*, ASTRAL, and ASTRID (all three trees computed by us), as well as to a concatenation analysis (obtained from [92]), with respect to branch support for established and proposed clades.

Experiment 6: Running time. We explore running time for SVDquartets+PAUP*, SVDquest, SVDquest*, and ASTRAL on the 37-taxon mammalian biological dataset.

7.3 Results

7.3.1 Results for Experiment 1

Experiment 1 compares SVDquest, SVDquest* and SVDquartets+PAUP* with respect to the MQSST scores they find. Although SVDquest does not always find better scores than SVDquartets+PAUP*, by design SVDquest* is guaranteed to find scores that are at least as large as those found by SVDquartets+PAUP*. In this section, we report the number of cases where SVDquest* finds a better score than SVDquartets+PAUP* and the number of cases where they have the same score; we also show the distribution of GTEE on the various datasets.

On the 50-taxon data, shown in Figure 7.1, a few basic trends are clear. SVDquest* has a much greater advantage over SVDquartets+PAUP* at higher ILS levels, almost always finding better scores on the highest ILS model condition. SVDquest* also has a larger advantage when there are 50 or 100 genes, as opposed to 500 or 1000 genes. Generally, the advantage of SVDquest*' over SVDquartets+PAUP* improves as gene tree estimation error (GTEE) increases, until the very highest GTEE rates where the advantage starts to

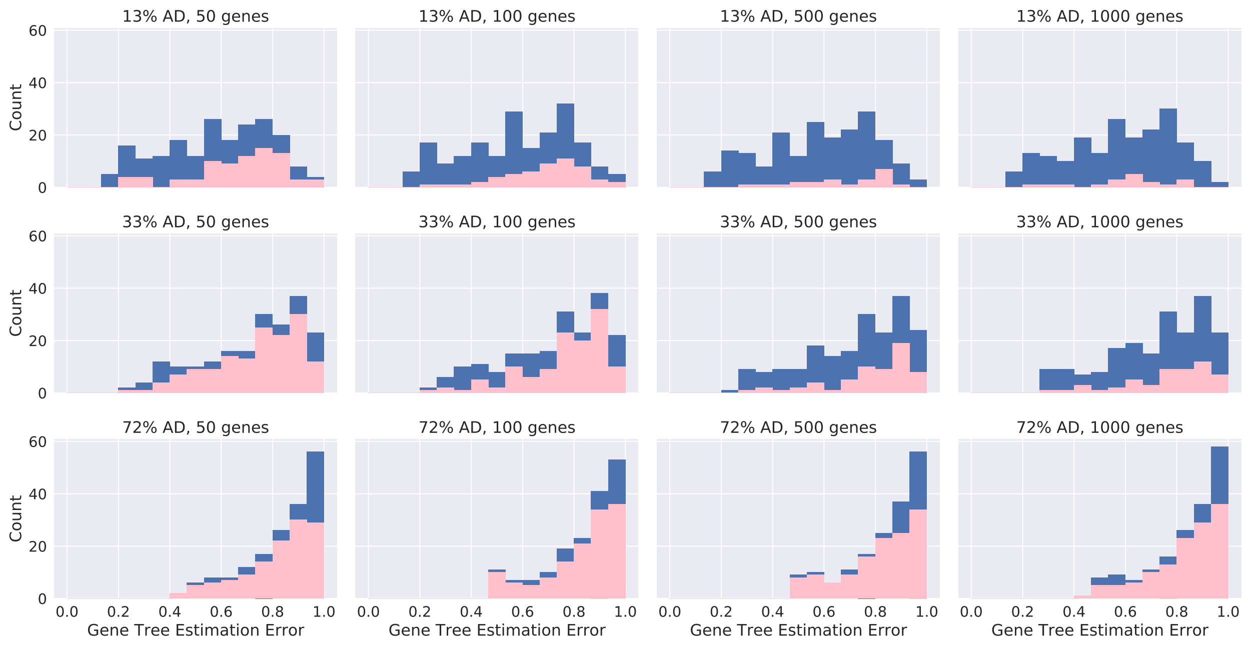


Figure 7.1: Results for Experiment 1 on 50-taxon data, showing how often SVDquest* finds better MQSST scores than SVDquartets+PAUP*. Pink sections of bars represent replicates where SVDquest* finds a better scoring tree; blue sections represent replicates where both methods find the same scoring tree. It is impossible for SVDquartets+PAUP* to find a better scoring tree than SVDquest*. Total heights of bars represent distribution of gene tree estimation error (maximum possible value is 1.0) in datasets. Each subfigure shows results for 200 replicates, with the exception of the AD=72% datasets, which have 195-200 replicates each.

fall. See also Supplementary Materials Figures S1-S3 for histograms of differences in MQSST scores between SVDquartets+PAUP* and SVDquest* on these datasets.

Results on the 15-taxon data (AD=82%), shown in Figure 7.2, also show that SVDquest* has a bigger advantage when there are fewer genes in the dataset. When there are 1000 genes, SVDquest* and SVDquartets+PAUP* almost always find trees with the same score, but SVDquest* frequently finds better trees when there are 50 or 100 genes. The impact of GTEE on the advantage with respect to MQSST score is less obvious on 50- and 100-gene datasets, but this may be because the range of GTEE is less than in the 50-taxon data.

The results on the 10-taxon data, shown in Figure 7.3, once again show that increasing the ILS level or decreasing the number of genes increases the frequency with which SVDquest finds a tree with a better MQSST score than SVDquartets+PAUP*. Like the 15-taxon datasets, which have similar levels of GTEE, the relationship between GTEE and the relative performance of the two methods on these datasets is less clear than on the 50-taxon datasets.

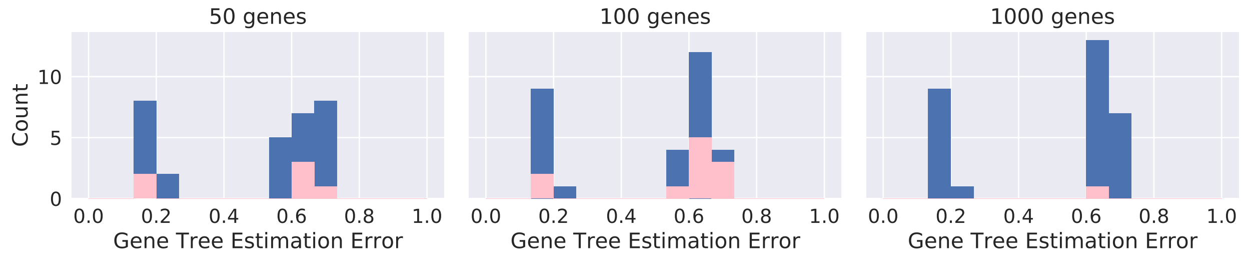


Figure 7.2: Results for Experiment 1 on 15-taxon data with 82% AD (high ILS), showing how often SVDquest* finds a better scoring tree than SVDquartets+PAUP*. Pink sections of bars represent replicates where SVDquest* finds a better scoring tree; blue sections represent replicates where both methods find the same scoring tree. It is impossible for SVDquartets+PAUP* to find a better scoring tree than SVDquest*. Total heights of bars represent distribution of gene tree estimation error (maximum possible value is 1.0) in datasets. Each subfigure shows results for 30 replicates.

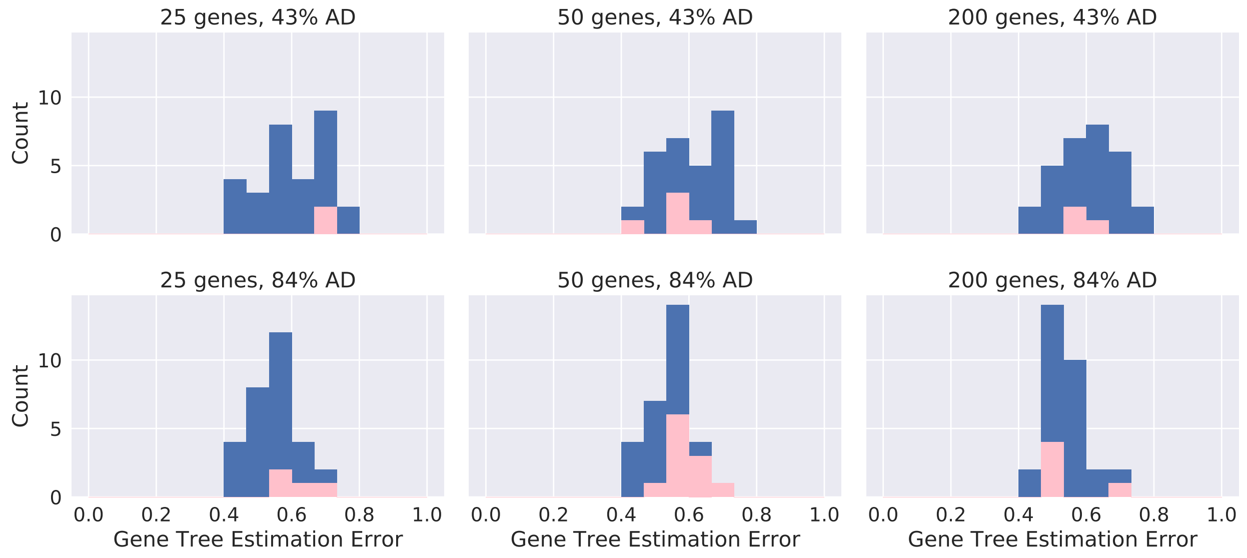


Figure 7.3: Results for Experiment 1 on 10-taxon data, showing how often SVDquest* finds a better scoring tree than SVDquartets+PAUP*. Pink sections of bars represent replicates where SVDquest* finds a better scoring tree; blue sections represent replicates where both methods find the same scoring tree. It is impossible for SVDquartets+PAUP* to find a better scoring tree than SVDquest*. Total heights of bars represent distribution of gene tree estimation error (maximum possible value is 1.0) in the datasets. Each subfigure shows results for 30 replicates.

7.3.2 Results for Experiment 2

Experiment 2 evaluates SVDquest*-strict (i.e., the strict consensus of all optimal trees found by SVDquest*, computed using SIESTA) in terms of topological accuracy on simulated c-gene datasets, and also compares it to ASTRID, ASTRAL, and SVDquartets+PAUP. The comparison between SVDquest*-strict and SVDquartets+PAUP* on the 50-taxon datasets (Supplementary Materials Figures S4-S7) shows that although SVDquest*-strict and SVDquartets+PAUP* have similar accuracy, SVDquartets+PAUP* has an advantage over SVDquartets+PAUP*.

A comparison between SVDquest*-strict, ASTRAL, and ASTRID on the 50-taxon datasets with 500 genes is shown in Figure 7.4 (see Supplementary Materials Figure S8 for other numbers of genes). These datasets do not evolve under a strict molecular clock and vary in ILS levels (reflected in AD percentages), GTEE, and number of genes. ASTRAL and ASTRID have similar accuracy levels under most conditions. At low levels of GTEE, all methods are fairly accurate. With high GTEE, SVDquest*-strict is much more accurate than ASTRAL and ASTRID. At high levels of ILS and low GTEE, ASTRAL and ASTRID are more accurate than SVDquest*-strict. Across all model conditions, the crossover point where SVDquest*-strict becomes more accurate is approximately 50% GTEE. SVDquest*-strict also has a bigger advantage over ASTRAL and ASTRID when ILS levels are lower and there are more genes. In the most extreme case with close to 100% GTEE, 13% AD, and 1000 genes (see Supplementary Materials, Figure S8), ASTRAL has approximately 75% estimation error while SVDquest*-strict has only 10% estimation error.

Figure 7.5 shows results on the 15-taxon datasets (AD=82%), which evolve under a strict molecular clock. ASTRAL is the most accurate method in all cases. The comparison between SVDquest*-strict and ASTRID shows that SVDquest*-strict has an advantage for the model conditions with largest number of genes (1000) and highest GTEE (40-60%), ASTRID has an advantage for the model conditions with fewest genes (50-100) and lowest GTEE (0-20%), and otherwise the two methods have similar species tree estimation error. However, this dataset has a relatively limited range of gene tree error - no replicate has greater than 60% average GTEE, which is the model condition where we would expect the best performance from SVDquest*-strict.

Results on the 10-taxon data (which do not evolve under a strict molecular clock) are shown in Figure 7.6. All three methods have similar levels of accuracy under most conditions. However, ASTRAL frequently returns slightly more topologically accurate trees than the other two methods, and ASTRAL and ASTRID are somewhat more accurate than SVDquest*-strict when there is low GTEE. Like the 15-taxon data, this model condition has no replicates with greater than 60% average GTEE.

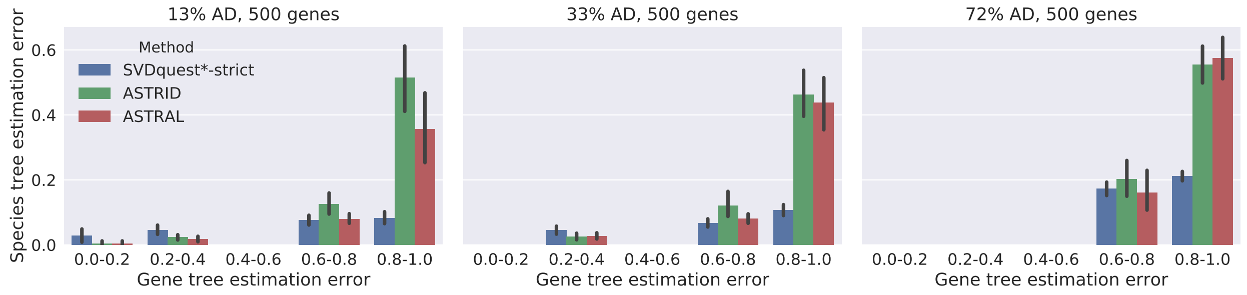


Figure 7.4: Species tree topological error rates (maximum possible is 1.0) for 50-taxon simulated data, as a function of percent gene tree estimation error (maximum possible is 1.0); the first two figures show results for 200 replicates and the last figure shows results for 198 replicates. Error bars show standard error.

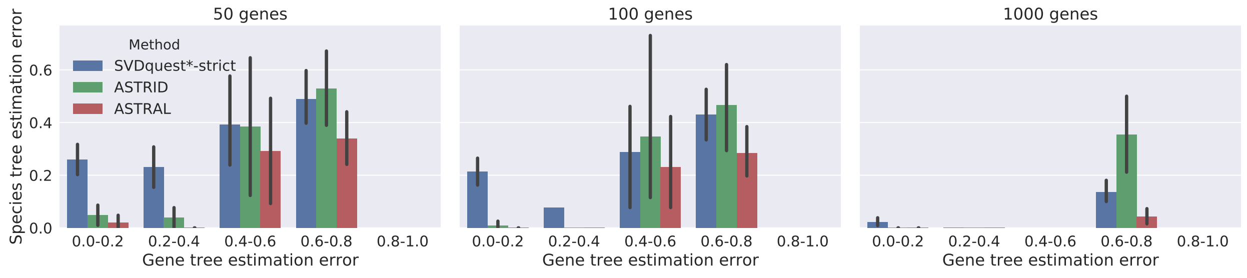


Figure 7.5: Species tree topological error rates (maximum possible is 1.0) for 15-taxon simulated data (AD=82%), as a function of gene tree estimation error (maximum possible is 1.0); each subfigure shows results on 30 replicates. Error bars show standard error.

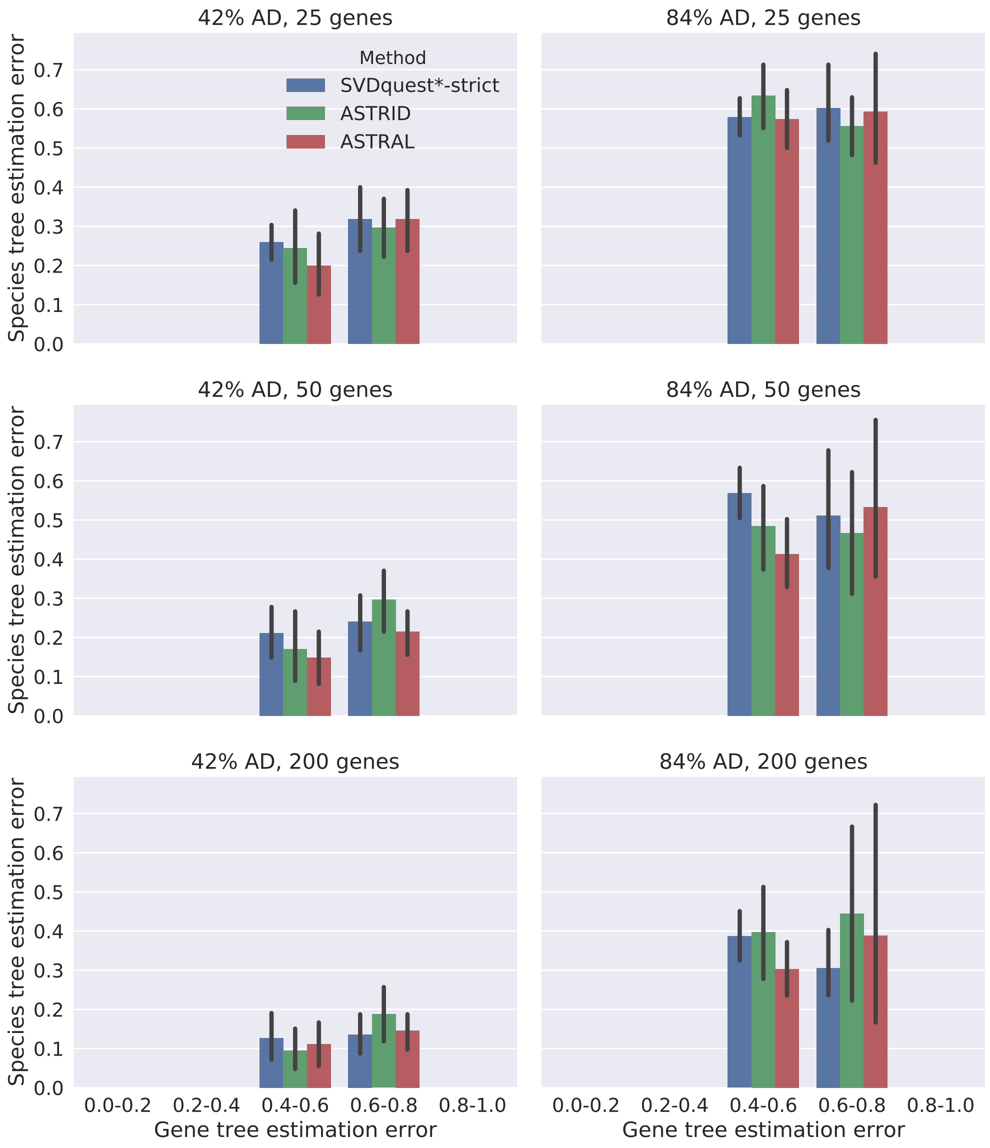


Figure 7.6: Species tree topological error rates (maximum possible is 1.0) for 10-taxon simulated data, as a function of gene tree estimation error (maximum possible is 1.0); each subfigure shows results for 30 replicates. Error bars show standard error.

7.3.3 Results for Experiment 3

Experiment 3 compares SVDquest*-strict to ASTRAL, ASTRID, and SVDquartets+PAUP* on supergene datasets (i.e., when loci are not recombination-free). We report both MQSST scores and topological accuracy.

The comparison between SVDquartets+PAUP* and SVDquest*-strict shows that SVDquest*-strict typically matches or improves on SVDquartets+PAUP* with respect to topological accuracy (Supplementary Materials, Figures S9-S11). In fact, the advantage of using SVDquest*-strict over SVDquartets+PAUP* is greater on the supergene datasets than on c-gene datasets. In what follows, we compare SVDquest*-strict to ASTRAL and ASTRID.

Results on the 50-taxon datasets are shown in Figure 7.7. The recombination-free loci have only 25 sites; all other lengths indicate supergenes obtained by combining c-genes. On all the model conditions with 25-site loci, SVDquest*-strict has a substantial advantage over ASTRID and ASTRAL. On the lowest ILS model condition, SVDquest*-strict retains the same accuracy as the c-genes are combined into supergenes. However, as the number of c-genes per supergene increases, ASTRAL and ASTRID become more accurate, eventually equaling or improving over SVDquest*-strict. For example, on the 33% AD model condition, SVDquest*-strict has an advantage when the supergenes have at most two c-genes, but then only ties with ASTRAL and ASTRID when there are more c-genes per supergene. On the 72% AD model condition, SVDquest*-strict retains an advantage regardless of the number of c-genes per supergene, but the advantage decreases with the length of the supergene.

Results on the 15-taxon datasets are seen in Figure 7.8. The c-genes have only 10 sites; all other lengths indicate supergenes obtained by binning together different c-genes. At the longest supergenes with 1000 sites (each composed of 100 recombination-free loci), ASTRAL and ASTRID find more accurate trees than SVDquest*-strict, but at lower levels of binning, SVDquest*-strict finds trees that are more accurate than ASTRID but less accurate than ASTRAL. ASTRAL finds slightly more accurate trees when the loci are recombination-free, while ASTRID improves substantially with increased binning, especially when there are 1000 loci. The impact of binning on SVDquest*-strict is minimal.

Relative performance on the 10-taxon data, shown in Figure 7.9, is similar to the 15-taxon data. ASTRAL typically becomes less accurate at higher levels of binning, while SVDquest*-strict is relatively unaffected, and ASTRID sometimes improves. These trends are more evident at the 84% AD level; at the 43% AD level, there is relatively little change with increased binning.

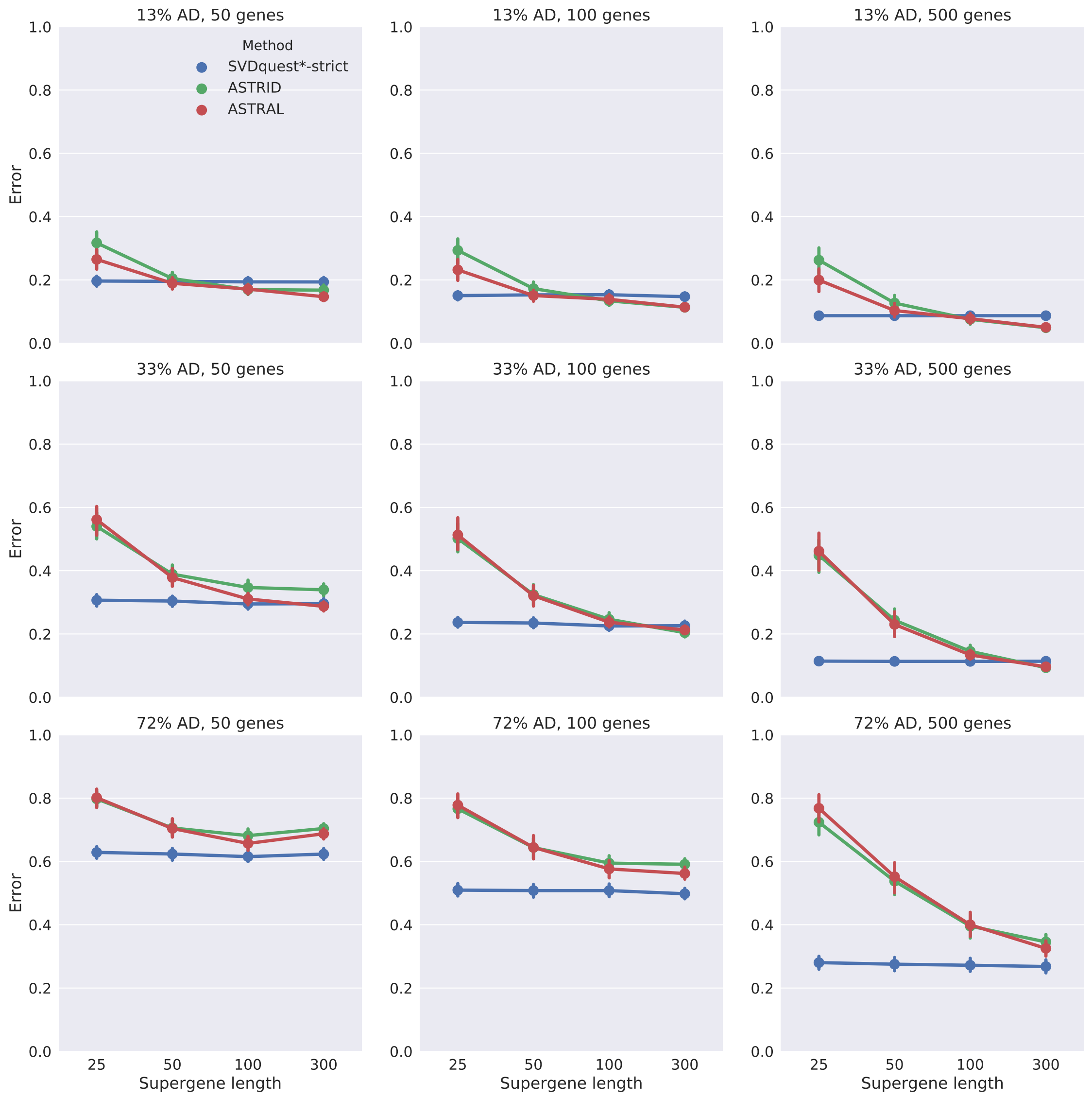


Figure 7.7: Species tree error rates (maximum possible is 1.0) on 50-taxon simulated data using supergenes (concatenations of c-genes) that may not be recombination-free. Error bars represent standard error of the mean. The c-genes in this experiment are 25 sites long, and multiple loci were concatenated to form supergenes. Thus, the 25-site genes have sites coming from one c-gene, the 50-site genes have sites coming from two c-genes, and the 100-site genes have sites coming from four c-genes. Each data point in a particular subfigure represents an analysis on the same number of sites. Each data point corresponds to an average over 50 replicates, except for the AD=72% 25-site 50-gene data point, which corresponds to 48 replicates, and 9 other AD=72% data points, each of which corresponds to 49 replicates.

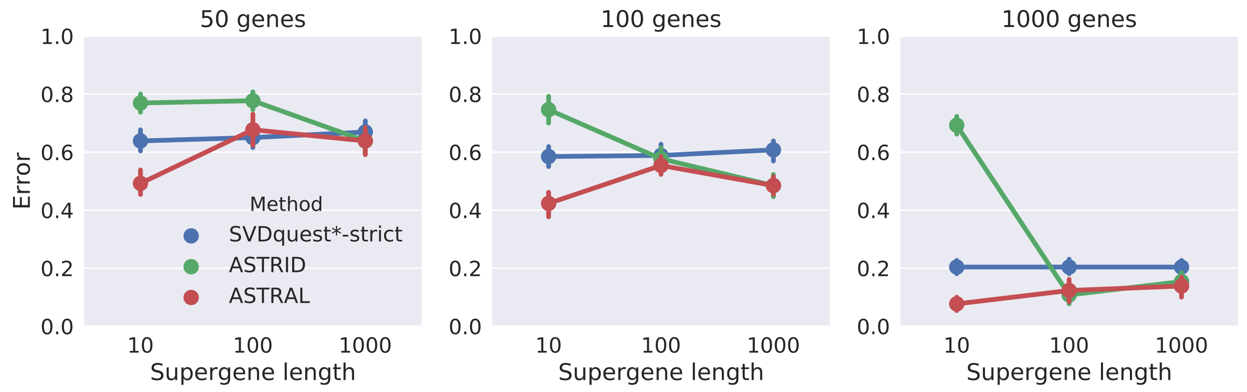


Figure 7.8: Species tree error rates (maximum possible is 1.0) on 15-taxon simulated data (AD=82%) for three different numbers of c-genes, then binned into supergenes (concatenations of c-genes); results shown are averaged over 10 replicates with error bars representing standard error of the mean. The c-genes in this experiment have 10 sites, so that longer loci are supergenes. Each data point in a particular subfigure represents an analysis on the same total number of sites.

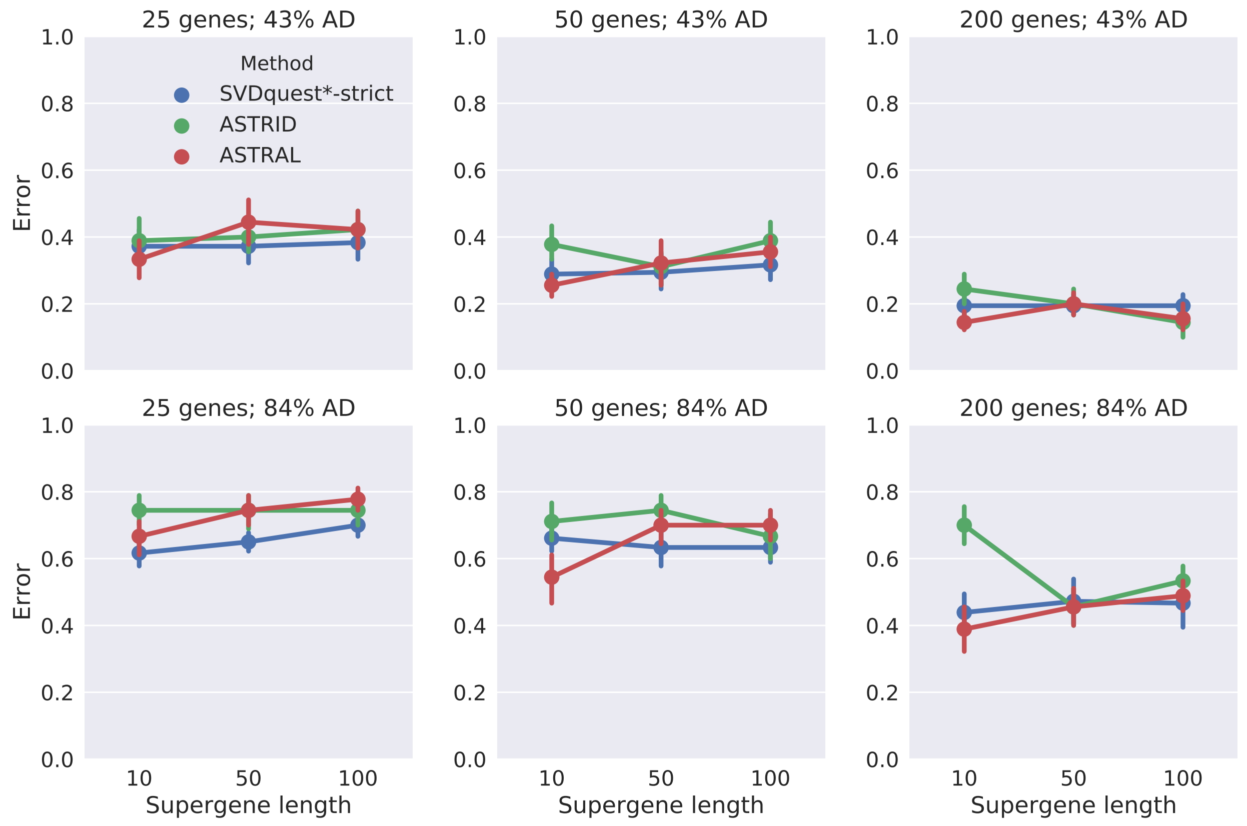


Figure 7.9: Species tree error rates (maximum possible is 1.0) on 10-taxon simulated data using supergenes (concatenations of c-genes), averaged over 10 replicates; error bars represent standard error of the mean. The c-genes in this experiment have 10 sites, and longer sequences are supergenes. Each data point in a particular subfigure represents an analysis on the same number of sites.

7.3.4 Results for Experiment 4

Experiment 4 compares coalescent-based methods to unpartitioned concatenation using RAxML (i.e., CA-ML) on simulated datasets. On the 50-taxon data, seen in Figure 7.10 for 500 gene datasets (see Supplementary Materials Figure S8 for other numbers of genes), CA-ML and SVDquest*-strict tend to perform similarly, and better than ASTRID and ASTRAL when GTEE is greater than 60%. On the lower ILS (13% AD) condition, CA-ML is somewhat more accurate than SVDquest*-strict when there are fewer genes (Supplementary Materials Figure S8), but this advantage is reduced for 500 or 1000 genes. At the highest ILS level, CA-ML is actually less accurate than SVDquest*-strict when there are few genes and low GTEE, but both of these methods are less accurate than ASTRAL and ASTRID.

On the 15-taxon data (AD=82%), shown in Figure 7.11, ASTRAL is always the most accurate method. ASTRID performs worse than CA-ML and SVDquest*-strict when there are 50 or 100 genes. With 1000 genes, all methods perform well, but ASTRAL and ASTRID slightly outperform CA-ML and SVDquest*-strict.

On the 10-taxon data, shown in Figure 7.12, CA-ML is typically the best method on the 43% AD data. CA-ML slightly outperforms the other methods except when there are 50 genes and low GTEE, in which case ASTRAL and ASTRID perform slightly better. On the 84% AD data, SVDquest*-strict and CA-ML are the worst performing methods, and ASTRAL is typically the best method.

We calculated the rank correlation coefficient between the MQSST scores and topological errors for PAUP* and SVDquest* in order to determine whether a better MQSST score was correlated with a topologically more accurate tree. We found that there was a statistically significant correlation ($P < 0.05$) with a Spearman rank correlation coefficient of $\rho = 0.32$ for the 50-taxon datasets.

7.3.4.1 Results for Experiment 5

We compared SVDquest* to SVDquartets+PAUP*, CA-ML, ASTRAL, and ASTRID trees on the mammalian biological dataset. CA-ML, ASTRAL, and ASTRID all return the same tree, which recovers the major accepted mammalian clades and the relationships between them, but SVDquest* returns a single tree that is different from the tree found by the other methods. See Figure 7.13 for the SVDquest* tree with bootstrap branch support, and Supplementary Materials Figure S12 for the ASTRAL/ASTRID/CA-ML tree with ASTRAL branch support. The SVDquest* tree has very high bootstrap branch support on nearly all the edges (100% support on all but four edges and 99% support on one edge), and the ASTRAL/CA-ML/ASTRID tree has over 90% support using ASTRAL's local posterior probability for all of its branches.

The SVDquest* tree agrees with SVDquartets+PAUP* but differs from the tree found using CA-ML, ASTRID, and ASTRID in two ways: the placement of tree shrews (Scandentia) and the topology of the clade Scrotifera with respect to the placement of bats (Chiroptera). CA-ML, ASTRAL and ASTRID place Scandentia as sister to Glires, while SVDquest* places Scandentia as sister to Primates. Both these placements have 100% support (bootstrap support for the SVDquest* tree, local support for the ASTRAL/CA-ML/ASTRID tree).

Scrotifera consists of three major clades - Chiroptera (bats), Cetartiodactyla (even-toed ungulates and cetaceans), and Zooamata (odd-toed ungulates and carnivores). CA-ML, ASTRAL and ASTRID resolve this clade with Chiroptera as the outgroup with 90% local support. SVDquest* resolves this with Zooamata as the outgroup, but with very low support (only 23% bootstrap support using the modified bootstrapping technique).

The existing literature presents varied hypotheses for Scrotifera. The SVDquest* analysis presents support for a clade that consists of Cetartiodactyla and Chiroptera, which has been presented by [50]. The CA-ML, ASTRAL, and ASTRID analyses present support for Fereuungulata, which contains Cetartiodactyla and Zooamata. More recent analyses [158] have found increased support for Fereuungulata over Cetartiodactyla+Chiroptera, but the phylogeny is not yet settled. However, the bootstrap support for Cetartiodactyla+Chiroptera in the SVDquest* tree is quite low (only 23%), and collapsing this edge makes the tree compatible with both of these two possibilities. SVDquest* establishes Zooamata with 69% support, which is only moderate. Collapsing edges in the SVDquest* tree with less than 75% bootstrap support resolves Cetartiodactyla, Chiroptera, Carnivora, and Perissodactyla as clades, but does not determine a relationship between them.

Finally, we also performed the standard non-parametric bootstrapping analysis on the SVDquest* tree, to evaluate the impact of using the modified bootstrapping technique for defining branch support. The results from the two techniques were nearly identical. All but three of the branches in the SVDquest* tree received exactly the same support using both techniques (91% for one branch, 100% for all the others). The differences in the support for the remaining three branches were very small. One branch that had 99% using the modified technique had 100% using the standard technique. The remaining two branches had support less than 75% using the modified bootstrapping technique, and their support changed by at most 5%: Ceteratiodactyla+Chiroptera received branch support of 23% using the modified technique and 19% using the standard technique, Zoomata received 69% support using the modified technique and 74% using the standard technique. Thus, the modified bootstrapping technique to provide branch support produces branch support values that are very close to that produced using the standard bootstrapping technique, while being much faster.

7.3.4.2 Experiment 6: Running Time

We compare the (sequential) running times of ASTRAL, SVDquest*, and SVDquartets+PAUP* on the 37-species mammalian dataset with 424 loci and a total of 1,338,678 characters. SVDquartets+PAUP* completed in under 4 minutes. ASTRAL and ASTRID both finished in just under 3.5 hours (less than one second difference), of which all but 4 seconds was spent computing ML gene trees. SVDquest* returned only one tree, and completed in under 3 hours and 32 minutes, which was just seconds more than what was needed to compute the ASTRAL and SVDquartets+PAUP* trees. The detailed running time analysis for SVDquest* is as follows:

- Computing maximum likelihood gene trees: 210 minutes
- Applying ASTRAL to the set of maximum likelihood gene trees, to obtain the constraint set of bipartitions: 6 seconds.
- Using PAUP* to compute SVDquartets quartet weights: 3 minutes.
- Applying PAUP* to the quartet trees to return the species tree: < 1 second.
- Running the dynamic programming within SVDquest* to find the optimal tree: 1 second.

Thus, the running time for SVDquest* (and for SVDquest*-strict) is essentially no different from that of running ASTRAL or ASTRID, and is dominated by the time used to compute ML gene trees. Also, SVDquartets+PAUP* is much faster than SVDquest* because SVDquartets+PAUP* does not need to compute gene trees.

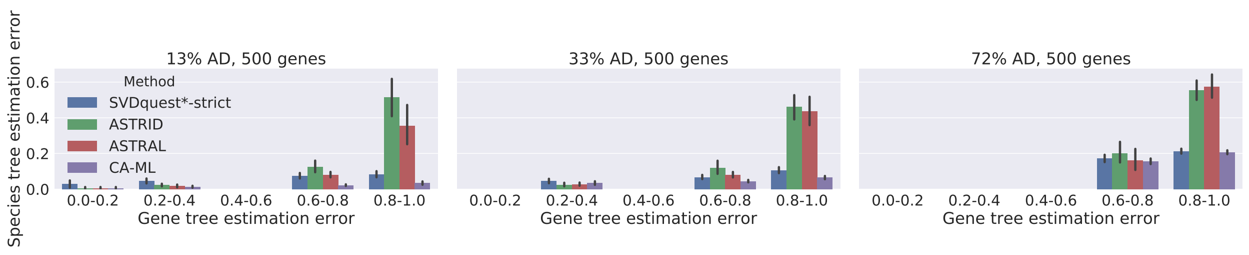


Figure 7.10: Species tree topological error rates (maximum possible is 1.0) for 50-taxon simulated data, as a function of gene tree estimation error (maximum possible is 1.0). Each figure shows means and standard error; results in the first two subfigures are for 200 replicates and 198 replicates for the third subfigure.

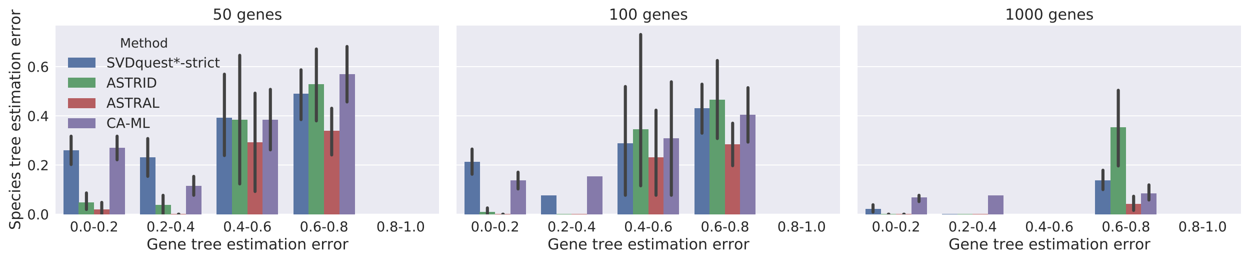


Figure 7.11: Species tree topological error rates (maximum possible is 1.0) for 15-taxon simulated data (AD=82%), as a function of gene tree estimation error (maximum possible is 1.0). Error bars show standard error over 10 replicates.

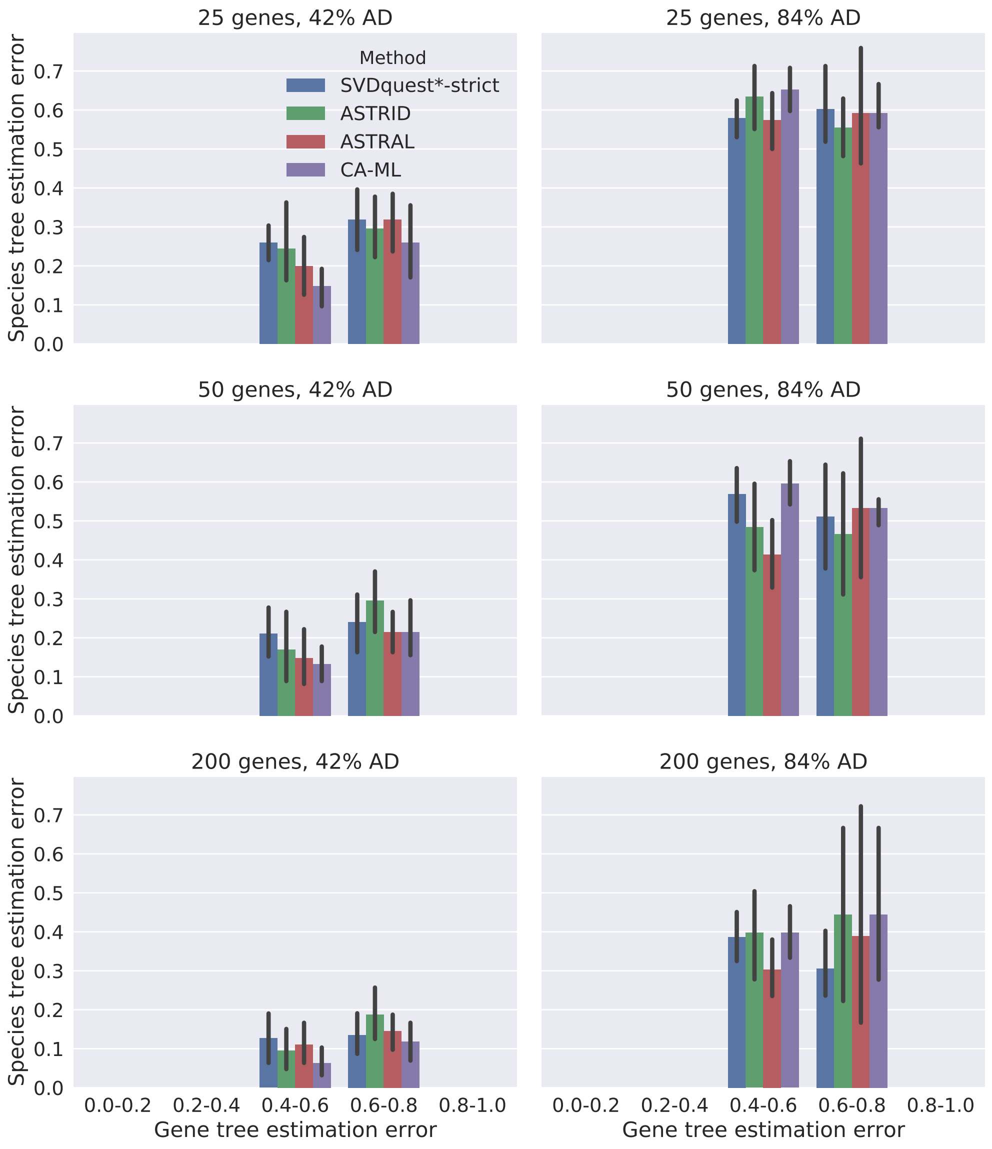


Figure 7.12: Species tree topological error rates (maximum possible is 1.0) for 10-taxon simulated data, as a function of gene tree estimation error (maximum possible is 1.0). Error bars show standard error over 10 replicates.

7.4 Discussion

MQSST scores. By design, it is impossible for SVDquest* to produce a tree with a worse MQSST score than SVDquartets+PAUP*. Hence, the question is how much better SVDquest* is than SVDquartets+PAUP* at finding good MQSST scores, and how the different model conditions affect the frequency with which SVDquest* improves on SVDquartets+PAUP*.

Our data show that SVDquest* often finds better scores than SVDquartets+PAUP*, but the frequency of this improvement depends on the model conditions, and is clearly related to the difficulty of the MQSST problem instance. Obviously, if SVDquartets+PAUP* finds an optimal solution, there is no better solution for SVDquest* to find. More generally, conditions that make it easy to find near-optimal MQSST using PAUP*'s heuristic search strategies will make it difficult for SVDquest* to do better than SVDquartets+PAUP*. ILS levels and number of genes both impact the relative performance, with an increasing advantage to SVDquest* over SVDquartets+PAUP* as ILS level increases or as the number of genes decreases. Both these trends are consistent with the hypothesis that easy conditions tend to reduce the advantage of SVDquest* over SVDquartets+PAUP* at finding good solutions to MQSST. The impact of GTEE is more complicated. Below about 30% GTEE, SVDquartets+PAUP* often finds a good MQSST score, so there is less room for SVDquest* to find an improvement. Above approximately 80% GTEE, SVDquartets+PAUP* might not find a good solution, but the gene trees have so much error that the constraint set computed by SVDquest* does not include parts of the solution space where better trees can be found. However, under other conditions (i.e., when GTEE is neither extremely low or extremely high), SVDquest* tends to produce better MQSST scores than SVDquartets+PAUP*.

These observations provide insights into the impact of GTEE on SVDquest*. It is well known that summary methods, such as ASTRAL and ASTRID, directly rely on estimated gene trees, and compute species trees based on summary statistics on the gene trees - quartet distributions or average internode distances. In contrast, gene tree estimation error only impacts how SVDquest* constrains the search space, and does not impact the criterion scores of any trees it can examine. Furthermore, the only real problem with using poorly estimated gene trees occurs when all the estimated gene trees are poor – because then the bipartitions of the true species tree may not end up in the constraint set. Adding bipartitions from poor gene trees to the constraint set expands the search space and hence increases the running time, but will never reduce the criterion score produced by SVDquest*. This suggests a general strategy of adding estimated species trees to the constraint set, even those that are not likely to be highly accurate; these expand the search space for SVDquest*, and are useful as long as they have a positive probability of containing a bipartition from a higher-scoring tree.

Species tree accuracy. A comparison between SVDquartets+PAUP* and SVDquest*-strict with respect to topological accuracy reveals that generally the differences are small, but that when the trees are different there is usually an improvement obtained by using SVDquest*-strict. The difference in accuracy is often small, but can be large (i.e., up to 10-15% in normalized RF). Hence, SVDquest*-strict provides an advantage (although slight) over SVDquartets+PAUP* in terms of species tree topology estimation.

The relative performance of ASTRAL and ASTRID in our study generally favored ASTRAL, in the sense that although the two methods were often very close in accuracy (and sometimes had identical accuracy), ASTRID was more impacted by GTEE than ASTRAL, and so was less accurate for the conditions with very short loci.

Both summary methods had very good accuracy – outperforming the other methods – when GTEE was sufficiently low and ILS was sufficiently high. However, CA-ML had the best accuracy under sufficiently low ILS levels, and even had the best accuracy under high ILS levels when GTEE was sufficiently high. SVDquest*-strict was less accurate than ASTRAL and ASTRID when GTEE was sufficiently low, but was as accurate as ASTRID and ASTRAL, and sometimes more accurate, when GTEE was very high.

Finally, although CA-ML typically dominated SVDquest*-strict, there were a few 10-taxon model conditions where SVDquest*-strict improved on the accuracy of CA-ML. Specifically, on the 10-taxon model conditions with high ILS (84% AD), high GTEE, and at least 50 genes, SVDquest*-strict was slightly more accurate than CA-ML.

Comparison to prior studies. Several other studies (surveyed in [100]) have compared coalescent-based methods and CA-ML under various simulated model conditions. These studies made the same general observations about the relative performance between the summary methods and CA-ML. Two prior studies [26, 100] have compared SVDquartets+PAUP* to other methods, including ASTRAL, NJst, ASTRID, and CA-ML; although SVDquartets+PAUP* sometimes improved over the summary methods when GTEE was sufficiently high, it was only rarely more accurate than CA-ML. Although we report results for SVDquest*-strict (which directly improves on SVDquartets+PAUP* for optimizing MQSST trees), our findings are also consistent with these general trends.

It is not clear what factors influence the relative accuracy of SVDquartets-based methods and CA-ML, although these studies as a whole suggest that when ILS is low enough, then CA-ML should be more accurate than SVDquest*-strict and SVDquartets+PAUP*. The total number of sites also seems to influence the relative performance, so that under high enough ILS and a large enough number of sites, SVDquartets-based methods may have an advantage over CA-ML. However, in our studies, when there was an advantage, it was small.

Experiment 3 suggests that species trees based on supergene trees (instead of on trees computed on c-genes) can sometimes improve the accuracy of species trees computed using SVDquest*-strict, as well as ASTRAL and ASTRID. The improvement for ASTRAL and ASTRID is consistent with a similar study (but applied to different summary methods) where supergenes are also based on random collections of genes [12]; furthermore, [65] also observed that coalescent-based summary methods were generally robust to recombination within loci. The improvement is perhaps surprising, since current theoretical justifications for using summary methods require that the loci be recombination-free. Furthermore, [128] argue that recombination-free loci may be extremely short (as few as 12 base pairs), and point out that on this basis the theoretical justification of summary methods is flawed. This concern is justified. However, from an empirical standpoint the results in these experiments suggest that failure to break loci into recombination-free regions may not be a substantial problem - and may even lead to improvements in some (but not all) cases.

Running time considerations. Our study also examined running time, and showed that SVDquest*-strict was reasonably fast. However, SVDquest* needs ASTRAL and SVDquartets+PAUP* to compute the constraint set, and so is necessarily more computationally intensive than both SVDquartets+PAUP* and ASTRAL. By far the dominant part of the running time for SVDquest*-strict is the gene tree estimation part, but this can be parallelized (i.e., each gene tree can be calculated independently of the others). In particular, it is feasible to run SVDquest*-strict on any dataset on which the full set of quartet trees can be computed using SVDquartets, which is also easily parallelized.

Future Work This study suggests multiple directions for future research. We used the default setting within PAUP* and we computed quartet trees for every four leaves; these choices are supposed to maximize the accuracy of SVDquartets+PAUP*, but it is possible that some other way of combining quartet trees within PAUP* would result in topologically more accurate trees. Similarly, quartet tree amalgamation is a basic algorithmic problem, and SVDquartets+PAUP* could be improved through the use of new quartet amalgamation methods. In addition, a branch-swapping heuristic could be developed that begins with the SVDquest* tree and searches for better solutions to MQSST; thus, SVDquartets+PAUP* can also be improved by incorporating SVDquest* as a starting tree. Furthermore, the basic strategy within SVDquest* of using other species tree methods to add bipartitions to the constraint set enables SVDquest to remain useful, even as PAUP* improves through the use of new quartet amalgamation heuristics.

Another interesting direction would be to modify the optimization problem that we solve. Thus, in the

MQSST problem, there is exactly one tree on every four species, and each of these quartet trees has unit weight. A weighted version of MQSST would be very interesting to examine, where instead of taking the best topology for each four taxa, the three possible topologies are weighted based on their SVD scores or on the statistical support for the quartet tree [39].

SVDquest*-strict could also be compared to PoMo [32] and its improved version revPoMo [122], which estimate species trees from multi-locus datasets under a model of site evolution that allows each node in the tree to be polymorphic. While these methods have not been shown to be statistically consistent under the MSC, they have shown very good accuracy on simulated data, even when gene tree heterogeneity due to ILS is present, and so may provide excellent accuracy in practice.

The accuracy of SVDquartets for computing quartet trees on biological datasets is not well understood, and this also presents multiple opportunities for future research. For example, this study examined the use of SVDquest* with multi-locus datasets, and assumed that gene trees can be computed on each of the loci. However, the basic algorithmic strategy in SVDquest can be used with SNP data as well, as we now describe. When the number of species is small enough (i.e., at most 20), then SVDquest could be used in its unconstrained mode: quartet trees can be computed using SVDquartets, and then a species tree optimizing the MQSST score can be found using the dynamic programming algorithm in SVDquest. For datasets with larger numbers of species, the constrained version can be used in several ways. For example, the constraint set can be initialized to the bipartitions in the SVDquartets+PAUP* tree, and then enlarged using standard CA-ML analyses, PoMo and revPomo (as described earlier), trees computed on bootstrap replicates, or other techniques. Similarly, our study examined SVDquest* on supergene datasets (formed by randomly concatenating c-genes) and showed good accuracy, but true recombination will produce patterns that are somewhat different, and the impact of recombination on SVDquest* and summary methods needs to be explored.

Another limitation of our study is that the simulations we performed evolved sequences only with substitutions (i.e., no insertions and deletions), and so alignment estimation was not necessary; yet alignment error is quite common in practice, especially when the datasets span large evolutionary timescales. Although alignment error also increases gene tree estimation error, several studies have shown that accurate gene trees can be computed even in the presence of some alignment error [71], so that it is possible that SVDquest* and other site-based methods could be more negatively impacted than summary methods. Hence, the impact of alignment error is an important aspect to consider. If alignment error negatively impacts SVDquartets, it may be that approaches that select sites within alignments to use within SVDquartets will be helpful.

Finally, although SVDquest*-strict is fast enough to be used on whole genome datasets with moderately large numbers of species, we only tested SVDquest*-strict under conditions where all quartet trees could be computed. Therefore, when the number of species is large enough (i.e., 200 or more), then this becomes computationally infeasible. For this reason, when the number of species is too large, PAUP* uses random sampling on the quartets, uses SVDquartets to compute quartet trees, and then combines these quartet trees using its quartet amalgamation heuristics. In its current implementation, SVDquest cannot be used with such inputs, but the dynamic programming algorithm in SVDquest can be used with any way of weighting quartet trees, and so could be used with sparsely sampled quartet trees by assigning equal weights to all three quartet trees on any unsampled quartet. However, sparse sampling of quartet trees for use with quartet amalgamation methods has been shown to have reduced accuracy compared to analyses that use all the quartet trees [135], suggesting that when the number of species makes SVDquest* inapplicable, summary methods or concatenation may be a better choice than SVDquartets-based approaches. Thus, the best modifications to SVDquest* to enable it to be used to good advantage on datasets with large numbers of species will require some investigation.

7.5 Summary

We presented SVDquest*, a site-based method for species tree estimation. Like the implementation within PAUP* (which we refer to as SVDquartets+PAUP*), SVDquest* operates by computing quartet trees using SVDquartets, and then seeks a species tree with the largest MQSST score. Unlike SVDquartets+PAUP*, which uses a heuristic search through treespace, SVDquest* uses an exact algorithm for this optimization problem, and achieves polynomial time by constraining the search space using a set of bipartitions on the species set that it computes from the input. By design, SVDquest* is guaranteed to obtain a score that is at least as large as the score produced using SVDquartets+PAUP*. In practice, SVDquest* typically finds trees with better MQSST scores than SVDquartets+PAUP*, especially on datasets with higher levels of gene tree estimation error and lower numbers of genes.

Our study evaluated SVDquest*-strict in comparison to two summary methods (ASTRAL and ASTRID), SVDquartets+PAUP*, and CA-ML under a wide range of ILS levels, numbers of species, and numbers of genes. Although our study was limited to conditions with at most 1000 genes and 50 species, we observed several significant and interesting trends. While ASTRAL and ASTRID can be more accurate than SVDquest*-strict when GTEE is low, SVDquest*-strict is typically more accurate than these summary methods when GTEE is high, as GTEE impacts summary methods directly, introducing error into the summary statistics they use to construct species trees. CA-ML is surprisingly accurate, and more accurate than the summary methods under conditions with high GTEE (even when ILS is high); interestingly, we also observed that sometimes SVDquest*-strict improves on CA-ML. Thus, the relative accuracy between these methods depends on the model condition, and in particular on the ILS and GTEE levels, but SVDquest*-strict provides advantages over the other coalescent-based methods under several biologically realistic conditions.

This study also shows that SVDquest*-strict is fast enough to use on genome-scale biological datasets. SVDquest*-strict includes calls to both ASTRAL and SVDquartets+PAUP*, and is otherwise very fast; hence, any dataset on which both of these methods can be run can be analyzed by SVDquest*-strict. Furthermore, a comparison of running times between these methods and concatenation suggests that for large enough datasets, concatenation analyses are likely to become computationally extremely expensive. For example, a concatenated maximum likelihood analysis of the 48-species avian phylogenomics dataset with 14,446 loci took more than 200 CPU years [53], while an analysis using the new implementation of ASTRAL took only 32 hours after the gene trees were computed [156]. The calculation of 14,446 ML gene trees is expensive, but completes in well under a month (and is very fast if parallelized) [53]. Hence, summary methods are generally computationally much more feasible than concatenation analyses for large datasets, which means that SVDquest*-strict is a computationally feasible approach for many genome-scale datasets.

This study adds to the current literature evaluating site-based approaches to species tree estimation. Although we did not find that SVDquest*-strict improved on the competing coalescent-based approaches under all conditions, our study shows that SVDquest*-strict can provide improved accuracy under some conditions with high GTEE. This trend suggests the potential for SVDquest*-strict to be particularly beneficial for genome-scale datasets, where GTEE is likely to be high as a result of either ILS or variable rates of evolution across the genome. In addition, SVDquest*-strict had very good accuracy on supergene datasets, suggesting it may be robust to failure to detect recombination events. Finally, the relative performance between SVDquest*-strict (and other methods based on SVDquartets), summary methods, and CA-ML might well depend on the number of loci, so that SVDquest*-strict (or other methods based on SVDquartets) could become the method of choice when the number of loci and ILS level are both very large.

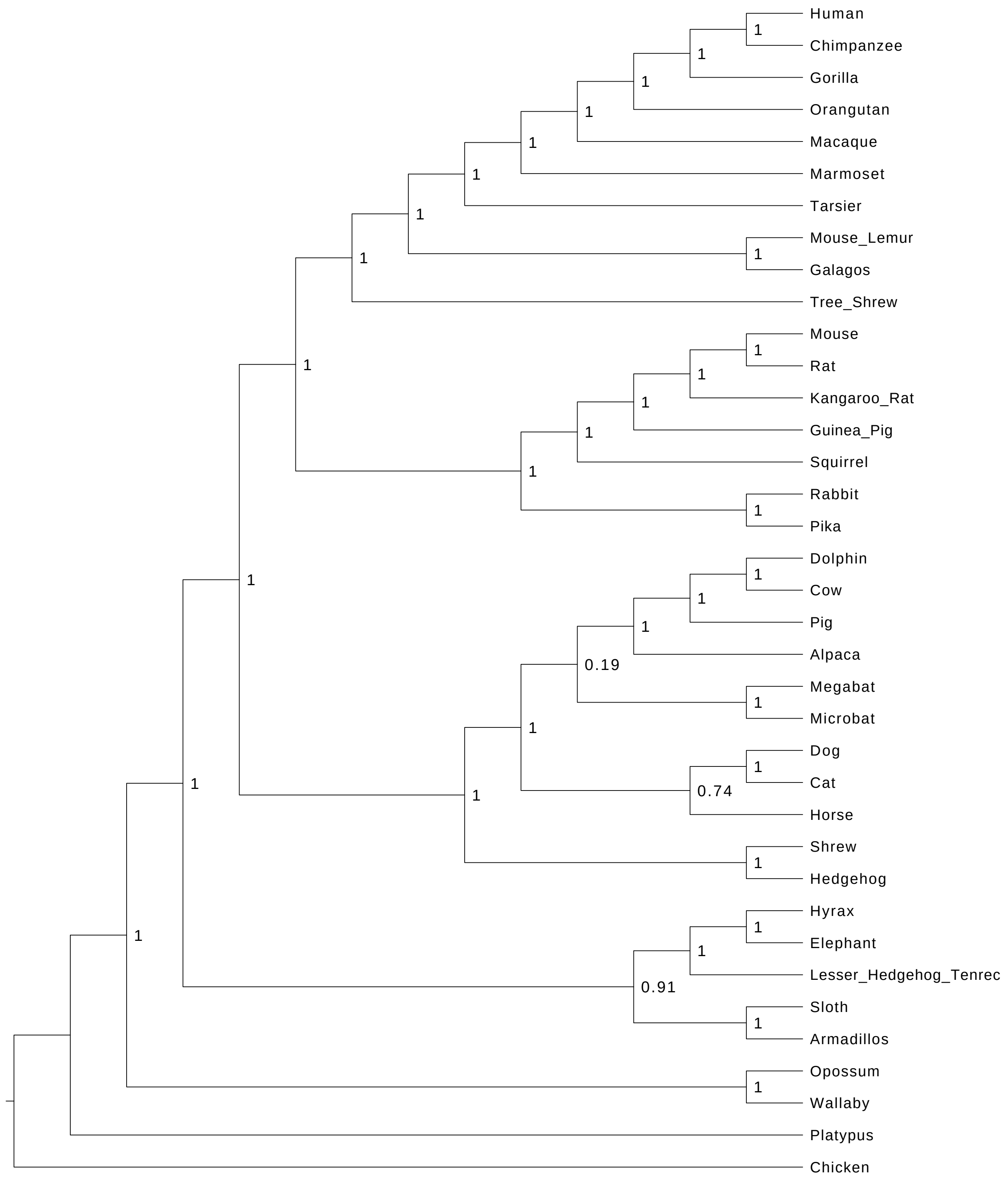


Figure 7.13: Mammalian SVDquest* tree with branch support, computed using a modified non-parametric bootstrapping approach, from 100 bootstrap replicates.

Chapter 8

Species Tree Estimation with ILS and HGT

Chapter 9

Conclusions

Chapter 10

Bibliography

Bibliography

[1] W.A. Akanni et al. “Horizontal gene transfer from Eubacteria to Archaebacteria and what it means for our understanding of eukaryogenesis”. In: *Phil. Trans. R. Soc. B* 370 (2015), p. 20140337.

[2] W.A. Akanni et al. “Implementing and testing Bayesian and maximum-likelihood supertree methods in phylogenetics”. In: *Royal Soc. open science* 2 (2015), p. 140436.

[3] W.A. Akanni et al. “L.U.-st: a tool for approximated maximum likelihood supertree reconstruction”. In: *BMC Bioinform.* 15 (2014), p. 183.

[4] Alana M Alexander et al. “Genomic data reveals potential for hybridization, introgression, and incomplete lineage sorting to confound phylogenetic relationships in an adaptive radiation of narrow-mouth frogs”. In: *Evolution* 71.2 (2017), pp. 475–488.

[5] Julie M Allen et al. “Phylogenomics from whole genome sequences using aTRAM”. In: *Systematic Biology* (2017), syw105.

[6] Elizabeth S Allman, James H Degnan, and John A Rhodes. “Split probabilities and species tree inference under the multispecies coalescent model”. In: *arXiv preprint arXiv:1704.04268* (2017).

[7] Diego F Alvarado-Serrano and Guillermo D’Elia. “A new genus for the Andean mice Akodon latebricola and A. bogotensis (Rodentia: Sigmodontinae)”. In: *Journal of Mammalogy* 94.5 (2013), pp. 995–1015.

[8] Benjamin M Anderson et al. “Genotyping-by-Sequencing in a Species Complex of Australian Hummock Grasses (Triodia): Methodological Insights and Phylogenetic Resolution”. In: *PloS one* 12.1 (2017), e0171053.

[9] Mukul S Bansal et al. “Robinson-Foulds supertrees”. In: *Algorithms for molecular biology* 5.1 (2010), p. 18.

[10] Bernard R. Baum. “Combining Trees as a Way of Combining Data Sets for Phylogenetic Inference, and the Desirability of Combining Gene Trees”. In: *Taxon* 41.1 (1992), pp. 3–10. ISSN: 00400262. URL: `http://www.jstor.org/stable/1222480`.

[11] Md Shamsuzzoha Bayzid, Siavash Mirarab, and Tandy J Warnow. “Inferring optimal species trees under gene duplication and loss.” In: *Pacific Symposium on Biocomputing*. Vol. 18. 2013, pp. 250–261.

[12] Md Shamsuzzoha Bayzid and Tandy Warnow. “Naive binning improves phylogenomic analyses”. In: *Bioinformatics* 29.18 (2013), pp. 2277–2284.

[13] Md Shamsuzzoha Bayzid et al. “Weighted statistical binning: enabling statistically consistent genome-scale phylogenetic analyses”. In: *PloS one* 10.6 (2015), e0129183.

[14] Md. S. Bayzid, T Hunt, and T. Warnow. “Disk covering methods improve phylogenomic analyses”. In: *BMC Genomics* 15 (Suppl 6) (2014), S7.

[15] R.M.D. Beck et al. “A higher-level MRP supertree of placental mammals”. In: *BMC Evolutionary Biology* 9.93 (2006).

[16] Ricardo Betancur-R and Guillermo Orti. “Molecular evidence for the monophyly of flatfishes (Carangimorpharia: Pleuronectiformes)”. In: *Molecular Phylogenetics and Evolution* 73 (2014), pp. 18–22.

[17] Olaf R.P. Bininda-Emonds. *Phylogenetic supertrees: combining information to reveal the "Tree of Life"*. Dordrecht: Springer Science & Business Media, 2004.

[18] FC Boucher et al. “Sequence capture using RAD probes clarifies phylogenetic relationships and species boundaries in Primula sect. Auricula”. In: *Molecular phylogenetics and evolution* 104 (2016), pp. 60–72.

[19] David Bryant and Mike Steel. “Computing the distribution of a tree metric”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 6.3 (2009), pp. 420–426.

[20] David Bryant and Mike Steel. “Constructing optimal trees from quartets”. In: *Journal of Algorithms* 38.1 (2001), pp. 237–259.

[21] David Bryant et al. “Inferring species trees directly from biallelic genetic markers: bypassing gene trees in a full coalescent analysis”. In: *Molecular biology and evolution* 29.8 (2012), pp. 1917–1932.

[22] Luke Charles Campillo. “Use of genomic data to resolve gene tree discordance in a Southeast Asian genus: Readdressing paraphyly in the spiderhunter (Nectariniidae, Arachnothera) phylogeny”. MA thesis. University of Kansas, 2016.

[23] M. Cardillo et al. “A species-level phylogenetic supertree of marsupials”. In: *Journal of Zoology* 264 (2004), pp. 11–31.

[24] Ruchi Chaudhary, David Fernández-Baca, and John Gordon Burleigh. “MulRF: a software package for phylogenetic analysis using multi-copy gene trees”. In: *Bioinformatics* 31.3 (2014), pp. 432–433.

[25] Julia Chifman and Laura Kubatko. "Quartet Inference from SNP Data Under the Coalescent Model". In: *Bioinformatics* (2014). DOI: 10.1093/bioinformatics/btu530. eprint: http://bioinformatics.oxfordjournals.org/content/early/2014/08/27/bioinformatics.btu530.full.pdf+html. URL: http://bioinformatics.oxfordjournals.org/content/early/2014/08/27/bioinformatics.btu530.abstract.

[26] J. Chou et al. "A comparative study of SVDquartets and other coalescent-based species tree estimation methods". In: *BMC Genomics* 16.Suppl 10 (2015), S2.

[27] Laura M Cisneros et al. "Phylogenetic supertree and functional trait database for all extant parrots". In: (2019).

[28] Alexis Criscuolo and Olivier Gascuel. "Fast NJ-like algorithms to deal with incomplete distance matrices". In: *BMC bioinformatics* 9.1 (2008), p. 166.

[29] Andrew A. Crowl, Cody Myers, and Nico Cellinese. "Embracing discordance: Phylogenomic analyses provide evidence for allopolyploidy leading to cryptic diversity in a Mediterranean Campanula (Campanulaceae) clade". In: *Evolution* 71.4 (2017), pp. 913–922. DOI: 10.1111/evo.13203.

[30] Charles Darwin. *On the origin of species.* John Murray, 1859.

[31] G Dasarathy, R Nowak, and S Roch. "Data Requirement for Phylogenetic Inference from Multiple Loci: A New Distance Method". In: *IEEE/ACM Trans Comp Biol Bioinformatics* 12 (2 2015). DOI: 10.1109/TCBB.2014.2361685, pp. 422–432.

[32] Nicola De Maio, Dominik Schrempf, and Carolin Kosiol. "PoMo: An Allele Frequency-Based Approach for Species Tree Estimation". In: *Systematic Biology* 64.6 (2015), pp. 1018–1031. DOI: 10.1093/sysbio/syv048. eprint: /oup/backfile/content_public/journal/sysbio/64/6/10.1093/sysbio/syv048/3/syv048.pdf. URL: +%20http://dx.doi.org/10.1093/sysbio/syv048.

[33] M. DeGiorgio and J. H. Degnan. "Robustness to divergence time underestimation when inferring species trees from estimated gene trees". In: *Syst. Biol.* 63.1 (2014), pp. 66–82.

[34] Michael DeGiorgio and James H Degnan. "Fast and consistent estimation of species trees using supermatrix rooted triples." In: *Molecular Biology and Evolution* 27.3 (Mar. 2010), pp. 552–69. ISSN: 1537-1719. DOI: 10.1093/molbev/msp250. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2877557%5C&tool=pmcentr%20ez%5C&rendertype=abstract.

[35] James H Degnan and Noah A Rosenberg. "Gene tree discordance, phylogenetic inference and the multispecies coalescent". In: *Trends in ecology & evolution* 24.6 (2009), pp. 332–340.

[36] Richard Desper and Olivier Gascuel. "Fast and accurate phylogeny minimum-evolution principle". In: *J Comput Biol* 9 (2002), pp. 687–705. ISSN: 1066-5277. DOI: 10.1089/106652702761034136.

[37] Markus Fleischauer and Sebastian Böcker. "Bad Clade Deletion supertrees: a fast and accurate supertree algorithm". In: *Molecular biology and evolution* 34.9 (2017), pp. 2408–2421.

[38] William Fletcher and Ziheng Yang. "INDELible: A Flexible Simulator of Biological Sequence Evolution". In: *Molecular Biology and Evolution* 26.8 (2009), pp. 1879–1888. DOI: 10.1093/molbev/msp098. eprint: http://mbe.oxfordjournals.org/content/26/8/1879.full.pdf+html. URL: http://mbe.oxfordjournals.org/content/26/8/1879.abstract.

[39] J. Gaither and L. Kubatko. "Hypothesis tests for phylogenetic quartets, with applications to coalescent-based species tree inference". In: *J. Theoretical Biology* 408 (2016), pp. 179–186.

[40] Olivier Gascuel. "BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data." In: *Molecular biology and evolution* 14.7 (1997), pp. 685–695.

[41] Olivier Gascuel and Mike Steel. "Neighbor-joining revealed". In: *Molecular biology and evolution* 23.11 (2006), pp. 1997–2000.

[42] J.P. Gatesy and M.S. Springer. "Phylogenetic analysis at deep timescales: Unreliable gene trees, bypassed hidden support, and the coalescence/concatalescence conundrum". In: *Mol Phylog Evol* 80 (2014), pp. 231–266.

[43] Raul E González-Ittig et al. "The molecular phylogenetics of the genus Oligoryzomys (Rodentia: Cricetidae) clarifies rodent host–hantavirus associations". In: *Zoological Journal of the Linnean Society* 171.2 (2014), pp. 457–474.

[44] Sheng Guo, Li-San Wang, and Junhyong Kim. "Large-scale simulation of RNA macroevolution by an energy-dependent fitness model". In: *arXiv preprint arXiv:0912.2326* (2009).

[45] Michael T Hallett and Jens Lagergren. "New algorithms for the duplication-loss model". In: *"Proceedings of the fourth annual International Conference on Computational Molecular Biology (RECOMB)"*. ACM. 2000, pp. 138–146.

[46] Kai He et al. "Talpid mole phylogeny unites shrew moles and illuminates overlooked cryptic species diversity". In: *Molecular Biology and Evolution* (2016), msw221.

[47] Joseph Heled and Alexei J Drummond. "Bayesian inference of species trees from multilocus data." In: *Molecular Biology and Evolution* 27.3 (Mar. 2010), pp. 570–80. ISSN: 1537-1719. DOI: 10.1093/molbev/msp274. URL: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2822290%5C&tool=pmcentr%20ez%5C&rendertype=abstract.

[48] Paul M Hime et al. “The influence of locus number and information content on species delimitation: an empirical test case in an endangered Mexican salamander”. In: *Molecular Ecology* 25.23 (2016), pp. 5959–5974.

[49] Peter A Hosner, Edward L Braun, and Rebecca T Kimball. “Rapid and recent diversification of curassows, guans, and chachalacas (Galliformes: Cracidae) out of Mesoamerica: Phylogeny inferred from mitochondrial, intron, and ultraconserved element sequences”. In: *Molecular Phylogenetics and Evolution* 102 (2016), pp. 320–330.

[50] Zhuo-cheng Hou, Roberto Romero, and Derek E Wildman. “Phylogeny of the Ferungulata (Mammalia: Laurasiatheria) as determined from phylogenomic data”. In: *Molecular phylogenetics and evolution* 52.3 (2009), pp. 660–664.

[51] Huateng Huang et al. “Sources of error inherent in species-tree estimation: impact of mutational and coalescent effects on accuracy and implications for choosing among different methods”. In: *Systematic Biology* 59.5 (2010), pp. 573–583.

[52] D. Huson, S. Nettles, and T. Warnow. “Disk-Covering, a fast converging method for phylogenetic tree reconstruction”. In: *Journal of Computational Biology* 6.3 (1999), pp. 369–386.

[53] Erich D Jarvis et al. “Whole-genome analyses resolve early branches in the tree of life of modern birds”. In: *Science* 346.6215 (2014), pp. 1320–1331.

[54] Tao Jiang, Paul Kearney, and Ming Li. “A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application”. In: *SIAM Journal on Computing* 30.6 (2001), pp. 1942–1961.

[55] Thomas H Jukes, Charles R Cantor, et al. “Evolution of protein molecules”. In: *Mammalian protein metabolism* 3.21 (1969), p. 132.

[56] Martyn Kennedy and Roderic DM Page. “Seabird supertrees: combining partial estimates of procellariiform phylogeny”. In: *The Auk* 119.1 (2002), pp. 88–108.

[57] J. F. C. Kingman. “On the Genealogy of Large Populations”. In: *Journal of Applied Probability* 19 (1982), p. 27. DOI: `10.2307/3213548`. URL: `http://www.jstor.org/stable/3213548`.

[58] L Lacey Knowles. “Estimating species trees: methods of phylogenetic analysis when there is incongruence across genes”. In: *Systematic Biology* 58.5 (2009), pp. 463–467.

[59] L.L. Knowles and L. Kubatko. *Estimating Species Trees: Practical and Theoretical Aspects.* Hoboken, NJ: Wiley-Blackwell, 2011.

[60] Klaus-Peter Koepfli et al. “The Genome 10K Project: a way forward”. In: *Annu. Rev. Anim. Biosci.* 3.1 (2015), pp. 57–111.

[61] L. Kubatko, B.C. Carstens, and L.L. Knowles. “ STEM: Species tree estimation using maximum likelihood for gene trees under coalescence”. In: *Bioinformatics* 25 (2009), pp. 971–973.

[62] Laura S Kubatko and J.H. Degnan. “Inconsistency of phylogenetic estimates from concatenated data under coalescence”. In: *Systematic Biology* 56 (2007), pp. 17–24.

[63] Anne Kupczok. “Split-based computation of majority-rule supertrees”. In: *BMC evolutionary biology* 11.1 (2011), p. 205.

[64] M. Lafond et al. “Gene Tree Construction and Correction using SuperTree and Reconciliation”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* PP.99 (2017), pp. 1–1. ISSN: 1545-5963. DOI: 10.1109/TCBB.2017.2720581.

[65] H.C. Lanier and L.L. Knowles. “Is recombination a problem for species-tree analyses?” In: *Systematic Biology* 61.4 (2012), pp. 691–701.

[66] Bret R Larget et al. “BUCKy: gene tree/species tree reconciliation with Bayesian concordance analysis”. In: *Bioinformatics* 26.22 (2010), pp. 2910–2911.

[67] Adam D Leaché et al. “Phylogenomics of phrynosomatid lizards: conflicting signals from sequence capture versus restriction site associated DNA sequencing”. In: *Genome Biology and Evolution* 7.3 (2015), pp. 706–719.

[68] Steven D Leavitt et al. “Resolving evolutionary relationships in lichen-forming fungi using diverse phylogenomic datasets and analytical approaches”. In: *Scientific reports* 6 (2016).

[69] Vincent Lefort, Richard Desper, and Olivier Gascuel. “FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program”. In: *Molecular biology and evolution* 32.10 (2015), pp. 2798–2800.

[70] Charles W Linkem, Vladimir N Minin, and Adam D Leaché. “Detecting the anomaly zone in species trees and evidence for a misleading signal in higher-level skink phylogeny (Squamata: Scincidae).” In: *Systematic biology* 65.3 (2016), pp. 465–477.

[71] Kevin Liu et al. “Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees”. In: *Science* 324.5934 (2009), pp. 1561–1564. DOI: 10.1126/science.1171243.

[72] Liang Liu. “BEST: Bayesian estimation of species trees under the coalescent model”. In: *Bioinformatics* 24.21 (2008), pp. 2542–2543. DOI: 10.1093/bioinformatics/btn484.

[73] Liang Liu and Dennis K Pearl. “Species trees from gene trees: reconstructing Bayesian posterior distributions of a species phylogeny using estimated gene tree distributions.” In: *Systematic Biology* 56.3 (June 2007), pp. 504–14. ISSN: 1063-5157. DOI: 10.1080/10635150701429982. URL: http://www.ncbi.nlm.nih.gov/pubmed/17562474.

[74] Liang Liu and Lili Yu. “Estimating species trees from unrooted gene trees”. In: *Systematic biology* 60.5 (2011), pp. 661–667.

[75] Liang Liu, Lili Yu, and Scott V Edwards. “A maximum pseudo-likelihood approach for estimating species trees under the coalescent model”. In: *BMC Evolutionary Biology* 10.1 (2010), p. 302. URL: http://www.biomedcentral.com/1471-2148/10/302.

[76] Liang Liu, Lili Yu, and Scott V Edwards. “A maximum pseudo-likelihood approach for estimating species trees under the coalescent model”. In: *BMC Evol Biol* 10 (2010), p. 302. ISSN: 1471-2148. DOI: 10.1186/1471-2148-10-302.

[77] Liang Liu et al. “Coalescent methods for estimating phylogenetic trees”. In: *Mol Phylogenet Evol* 53.1 (2009), pp. 320–328.

[78] Liang Liu et al. “Estimating species phylogenies using coalescence times among sequences”. In: *Systematic Biology* 58.5 (2009), pp. 468–477.

[79] Leonardo F Machado et al. “Phylogeny and biogeography of tetralophodont rodents of the tribe Oryzomyini (Cricetidae: Sigmodontinae)”. In: *Zoologica Scripta* 43.2 (2014), pp. 119–130.

[80] Wayne P Maddison. “Gene trees in species trees”. In: *Systematic biology* 46.3 (1997), pp. 523–536.

[81] Renan Maestri et al. “The ecology of a continental evolutionary radiation: Is the radiation of sigmodontine rodents adaptive?” In: *Evolution* 71.3 (2017), pp. 610–632.

[82] D. Mallo and D. Posada. “Multilocus inference of species trees and DNA barcoding”. In: *Phil. Trans. R. Soc. B* 371.1702 (2016), p. 20150335.

[83] Diego Mallo, Leonardo De Oliveira Martins, and David Posada. “SimPhy: phylogenomic simulation of gene, locus, and species trees”. In: *Systematic biology* 65.2 (2016). 10.1093/sysbio/syv082, pp. 334–344.

[84] Joseph D Manthey, Mark Geiger, and Robert G Moyle. “Relationships of morphological groups in the northern flicker superspecies complex (Colaptes auratus & C. chrysoides)”. In: *Systematics and Biodiversity* 15.3 (2017), pp. 183–191.

[85] Joseph D Manthey et al. “Comparison of target-capture and restriction-site associated DNA sequencing for phylogenomics: a test in cardinalid tanagers (Aves, Genus: Piranga)”. In: *Systematic biology* (2016), syw005.

[86] John E. McCormack et al. “A Phylogeny of Birds Based on Over 1,500 Loci Collected by Target Enrichment and High-Throughput Sequencing”. In: *PLoS One* 8.1 (2013), e54848.

[87] M. McMahon and M Sanderson. “Phylogenetic supermatrix analysis of GenBank sequences from 2228 papilionoid legumes”. In: *Systematic Biology* 55 (2006), pp. 818–836.

[88] Michelle M McMahon and Michael J Sanderson. “Phylogenetic supermatrix analysis of GenBank sequences from 2228 papilionoid legumes”. In: *Systematic biology* 55.5 (2006), pp. 818–836.

[89] K.A. Meiklejohn et al. “Analysis of a rapid evolutionary radiation using ultraconserved elements: evidence for a bias in some multispecies coalescent methods”. In: *Systematic Biology* 65.4 (2016). doi: 10.1093/sysbio/syw014, pp. 612–627.

[90] Kelly A Meiklejohn et al. “Analysis of a rapid evolutionary radiation using ultraconserved elements: evidence for a bias in some multispecies coalescent methods”. In: *Systematic biology* 65.4 (2016), pp. 612–627.

[91] S. Mir Arabbaygi. “Novel Scalable Approaches for Multiple Sequence Alignment and Phylogenomic Reconstruction”. PhD thesis. The University of Texas at Austin, 2015.

[92] S. Mirarab et al. “ASTRAL: Accurate Species TRee ALgorithm”. In: *Bioinformatics* 30.17 (2014), pp. i541–i548.

[93] Siavash Mirarab, Md Shamsuzzoha Bayzid, and Tandy Warnow. “Evaluating summary methods for multilocus species tree estimation in the presence of incomplete lineage sorting”. In: *Systematic biology* 65.3 (2014), pp. 366–380.

[94] Siavash Mirarab and Tandy Warnow. “ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes”. In: *Bioinformatics* 31.12 (2015), pp. i44–i52.

[95] Siavash Mirarab et al. “ASTRAL: genome-scale coalescent-based species tree estimation”. In: *Bioinformatics* 30.17 (2014), pp. i541–i548.

[96] Siavash Mirarab et al. “PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences”. In: *Journal of Computational Biology* 22.5 (2015), pp. 377–386.

[97] Siavash Mirarab et al. “Statistical binning enables an accurate coalescent-based estimation of the avian tree”. In: *Science* 346.6215 (2014), p. 1250463.

[98] Siavash Mirarab et al. "Statistical binning improves species tree estimation in the presence of gene tree incongruence". In: *Science* (2014). (Under review, companion paper for Avian Phylogenomics project).

[99] Nora Mitchell et al. "Anchored phylogenomics improves the resolution of evolutionary relationships in the rapid radiation of Protea L." In: *American Journal of Botany* 104.1 (2017), pp. 102–115. DOI: 10.3732/ajb.1600227. eprint: http://www.amjbot.org/content/104/1/102.full.pdf+html. URL: http://www.amjbot.org/content/104/1/102.abstract.

[100] Erin K. Molloy and Tandy Warnow. "To Include or Not to Include: The Impact of Gene Filtering on Species Tree Estimation Methods". In: *Systematic Biology* (2017). DOI: https://doi.org/10.1093/sysbio/syx077.

[101] E. Mossel and S. Roch. "Distance-based species tree estimation: information-theoretic trade-off between number of loci and sequence length under the coalescent". In: *arXiv preprint* (2015). arXiv:1504.05289v1.

[102] Elchanan Mossel and Sebastien Roch. "Incomplete lineage sorting: consistent phylogeny estimation from multiple loci". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 7.1 (2010), pp. 166–171.

[103] Robert G Moyle et al. "Tectonic collision and uplift of Wallacea triggered the global songbird radiation". In: *Nature Communications* 7 (2016).

[104] Serita Nelesen et al. "DACTAL: divide-and-conquer trees (almost) without alignments". In: *Bioinformatics* 28.12 (2012), pp. i274–i282.

[105] Nam Nguyen, Siavash Mirarab, and Tandy Warnow. "MRL and SuperFine+MRL: new supertree methods". In: *Algorithms for Molecular Biology* 7.1 (2012), p. 3.

[106] Adrián Nieto-Montes de Oca et al. "Phylogenomics and species delimitation in the knob-scaled lizards of the genus Xenosaurus (Squamata: Xenosauridae) using ddRADseq data reveal a substantial underestimation of diversity". In: *Molecular Phylogenetics and Evolution* 106 (2017), pp. 241–253.

[107] S. Ohno. *Evolution by Gene Duplication*. New York: Springer-Verlag, 1970.

[108] Swati Patel, Rebecca T Kimball, and Edward L Braun. "Error in phylogenetic estimation for bushes in the tree of life". In: *Journal of Phylogenetics & Evolutionary Biology* (2013).

[109] Morgan N Price, Paramvir S Dehal, and Adam P Arkin. "FastTree 2–approximately maximum-likelihood trees for large alignments". In: *PloS one* 5.3 (2010), e9490.

[110] Mark A Ragan. "Phylogenetic inference based on matrix representation of trees". In: *Molecular phylogenetics and evolution* 1.1 (1992), pp. 53–58.

[111] Benjamin D. Redelings and Mark T. Holder. "A supertree pipeline for summarizing phylogenetic and taxonomic information for millions of species". In: *PeerJ* 5 (Mar. 2017), e3058. ISSN: 2167-8359. DOI: `10.7717/peerj.3058`. URL: `https://doi.org/10.7717/peerj.3058`.

[112] David F Robinson and Leslie R Foulds. "Comparison of phylogenetic trees". In: *Mathematical biosciences* 53.1 (1981), pp. 131–147.

[113] S Roch. "Towards extracting all phylogenetic information from matrices of evolutionary distances". In: *Science* 327.5971 (2010), pp. 1376–1379.

[114] Sebastien Roch. "A short proof that phylogenetic tree reconstruction by maximum likelihood is hard". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 3.1 (2006), p. 92.

[115] Sebastien Roch and Mike Steel. "Likelihood-based tree reconstruction on a concatenation of aligned sequence data sets can be statistically inconsistent". In: *Theoretical population biology* 100 (2015), pp. 56–62.

[116] Sebastien Roch and Mike Steel. "Likelihood-based tree reconstruction on a concatenation of alignments can be statistically inconsistent". In: *Theoretical Population Biology* 100 (2015), pp. 56–62.

[117] Sebastien Roch and Tandy Warnow. "On the Robustness to Gene Tree Estimation Error (or lack thereof) of Coalescent-Based Species Tree Methods". In: *Systematic Biology* 64.4 (2015), pp. 663–676.

[118] Fredrik Ronquist et al. "MrBayes 3.2: efficient Bayesian phylogenetic inference and model choice across a large model space". In: *Systematic biology* 61.3 (2012), pp. 539–542.

[119] Carl J Rothfels et al. "The evolutionary history of ferns inferred from 25 low-copy nuclear genes". In: *American Journal of Botany* 102.7 (2015), pp. 1089–1107.

[120] Naruya Saitou and Masatoshi Nei. "The neighbor-joining method: a new method for reconstructing phylogenetic trees." In: *Molecular biology and evolution* 4.4 (1987), pp. 406–425.

[121] Erfan Sayyari and Siavash Mirarab. "Fast coalescent-based computation of local branch support from quartet frequencies". In: *Molecular biology and evolution* 33.7 (2016), pp. 1654–1668.

[122] Dominik Schrempf et al. "Reversible polymorphism-aware phylogenetic models and their application to tree inference". In: *Journal of Theoretical Biology* 407.Supplement C (2016), pp. 362–370. ISSN: 0022-5193. DOI: `https://doi.org/10.1016/j.jtbi.2016.07.042`. URL: `http://www.sciencedirect.com/science/article/pii/S0022519316302259`.

[123] Barbara J Sharanowski et al. "Expressed sequence tags reveal Proctotrupomorpha (minus Chalcidoidea) as sister to Aculeata (Hymenoptera: Insecta)". In: *Molecular Phylogenetics and Evolution* 57.1 (2010), pp. 101–112.

[124] Martin Simonsen, Thomas Mailund, and Christian NS Pedersen. "Inference of large phylogenies using neighbour-joining". In: *International Joint Conference on Biomedical Engineering Systems and Technologies*. Springer. 2010, pp. 334–344.

[125] Martin Simonsen, Thomas Mailund, and Christian NS Pedersen. "Rapid neighbour-joining". In: *International Workshop on Algorithms in Bioinformatics*. Springer. 2008, pp. 113–122.

[126] Robert R Sokal. "A statistical method for evaluating systematic relationship". In: *University of Kansas science bulletin* 28 (1958), pp. 1409–1438.

[127] Sen Song et al. "Resolving conflict in eutherian mammal phylogeny using phylogenomics and the multispecies coalescent model". In: *Proceedings of the National Academy of Sciences* 109.37 (2012), pp. 14942–14947.

[128] M.S. Springer and J. Gatesy. "The gene tree delusion". In: *Molecular Phylogenetics and Evolution* 94 (2016), pp. 1–33.

[129] Alexandros Stamatakis. "RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies". In: *Bioinformatics* 30.9 (2014), pp. 1312–1313.

[130] Alexandros Stamatakis. "RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models." In: *Bioinformatics* 22.21 (2006), pp. 2688–2690. URL: `http://www.ncbi.nlm.nih.gov/pubmed/16928733`.

[131] Mike Steel and Allen Rodrigo. "Maximum likelihood supertrees". In: *Systematic biology* 57.2 (2008), pp. 243–250.

[132] Jeet Sukumaran and Mark T Holder. "DendroPy: a Python library for phylogenetic computing". In: *Bioinformatics* 26.12 (2010), pp. 1569–1571.

[133] M Shel Swenson et al. "A simulation study comparing supertree and combined analysis methods using SMIDGen". In: *Algorithms for Molecular Biology* 5.1 (2010), p. 8.

[134] M Shel Swenson et al. "SuperFine: fast and accurate supertree estimation". In: *Systematic biology* 61.2 (2011), p. 214.

[135] M. Shel Swenson et al. "An experimental study of Quartets MaxCut and other supertree methods". In: *Algorithms for Molecular Biology* 6.1 (Apr. 2011), p. 7. ISSN: 1748-7188. DOI: `10.1186/1748-7188-6-7`. URL: `https://doi.org/10.1186/1748-7188-6-7`.

[136] D Swofford. *PAUP*. Phylogenetic Analysis Using Parsimony (*and other methods). Version 4.* Sunderland, Massachussets: Sinauer Associates, 2003.

[137] Gergely J Szöllősi et al. "Efficient exploration of the space of reconciled gene trees". In: *Systematic biology* (2013), syt054.

[138] Cuong Q Tang et al. "Effects of phylogenetic reconstruction method on the robustness of species delimitation using single-locus data". In: *Methods in Ecology and Evolution* 5.10 (2014), pp. 1086–1094.

[139] C Than and L Nakhleh. "Species tree inference by minimizing deep coalescences". In: *PLoS Computational Biology* 5.9 (2009), e1000501. DOI: `10.1371/journal.pcbi.1000501`. URL: `http://dx.plos.org/10.1371/journal.pcbi.1000501.g016`.

[140] Pranjal Vachaspati. *Simulated Data for SIESTA paper.* Retrieved July 21, 2017 from doi:10.6084/m9.figshare.5234803.v 2017. DOI: `10.6084/m9.figshare.5234803.v1`.

[141] Pranjal Vachaspati and Tandy Warnow. "ASTRID: accurate species trees from internode distances". In: *BMC genomics* 16.10 (2015), S3.

[142] Pranjal Vachaspati and Tandy Warnow. "Enhancing Searches for Optimal Trees Using SIESTA". In: *Proceedings of the 15th International Workshop, RECOMB -Comparative Genomics, Barcelona, Spain, October 4-6, 2017.* Ed. by Joao Meidanis and Luay Nakhleh. Springer International Publishing, 2017, pp. 232–255. DOI: `10.1007/978-3-319-67979-2_13`.

[143] Pranjal Vachaspati and Tandy Warnow. "Enhancing searches for optimal trees using SIESTA". In: *RECOMB International Workshop on Comparative Genomics.* Springer. 2017, pp. 232–255.

[144] Pranjal Vachaspati and Tandy Warnow. "FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization". In: *Bioinformatics* 33.5 (2017), pp. 631–639.

[145] T. Warnow, B. M. E. Moret, and K. St. John. "Absolute phylogeny: true trees from short sequences". In: *Proc. 12th Ann. ACM/SIAM Symp. on Discr. Algs. SODA01.* SIAM Press, 2001, pp. 186–195.

[146] Tandy Warnow. “Divide-and-conquer tree estimation: opportunities and challenges”. In: *Bioinformatics and Phylogenetics*. Springer, 2019, pp. 121–150.

[147] Tandy Warnow. “Supertree construction: Opportunities and challenges”. In: *arXiv preprint arXiv:1805.03530* (2018).

[148] Noor D White, Charles Mitter, and Michael J Braun. “Ultraconserved Elements Resolve the Phylogeny of Potoos (Aves: Nyctibiidae)”. In: *Journal of Avian Biology* (2017). doi: 10.111/jav.01313.

[149] Noor D White et al. “A multi-gene estimate of higher-level phylogenetic relationships among nightjars (AVES: CAPRIMULGIDAE)”. In: *Ornitologia Neotropical* 27 (2016), pp. 223–236.

[150] Norman J Wickett et al. “Phylotranscriptomic analysis of the origin and early diversification of land plants”. In: *Proceedings of the National Academy of Sciences* 111.45 (2014), E4859–E4868.

[151] Carl R Woese. “On the evolution of cells”. In: *Proceedings of the National Academy of Sciences* 99.13 (2002), pp. 8742–8747.

[152] M. Wojciechowski et al. “Molecular phylogeny of the “temperate herbaceous tribes” of papilionoid legumes: a supertree approach”. In: *Advances in Legume Systematics* 9 (2000), pp. 277–298.

[153] Martin F Wojciechowski et al. “Molecular phylogeny of the “temperate herbaceous tribes” of papilionoid legumes: a supertree approach”. In: *Advances in legume systematics* 9 (2000), pp. 277–298.

[154] John Yin, Chao Zhang, and Siavash Mirarab. “ASTRAL-MP: scaling ASTRAL to very large datasets using randomization and parallelization”. In: *Bioinformatics* (2019).

[155] Yun Yu, Tandy Warnow, and Luay Nakhleh. “Algorithms for MDC-based multi-locus phylogeny inference: beyond rooted binary gene trees on single alleles”. In: *Journal of Computational Biology* 18.11 (2011), pp. 1543–1559.

[156] C. Zhang, E. Sayyari, and S. Mirarab. “ASTRAL-III: Increased Scalability and Impacts of Contracting Low Support Branches”. In: *Comparative Genomics: Proceedings of the 15th International Workshop, RECOMB-CG 2017, Barcelona, Spain, October 4-6, 2017*. Ed. by J. Meidanis and L. Nakhleh. Cham, Switzerland: Springer International Publishing, 2017, pp. 53–75.

[157] Guojie Zhang. “Genomics: Bird sequencing project takes off”. In: *Nature* 522.7554 (2015), p. 34.

[158] Xuming Zhou et al. “Phylogenomic Analysis Resolves the Interordinal Relationships and Rapid Diversification of the Laurasiatherian Mammals”. In: *Systematic Biology* 61.1 (2012), p. 150.

[159] T. Zimmermann, S. Mirarab, and T. Warnow. “BBCA: Improving the scalability of *BEAST using random binning”. In: *BMC Genomics* 15 (Suppl 6) (2014), S11.