SUPERTREE-LIKE METHODS FOR GENOME-SCALE SPECIES TREE ESTIMATION

BY

ERIN MOLLOY

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, 2020

Urbana, Illinois

Doctoral Committee:

Professor Tandy Warnow, Chair
Professor William Gropp, Co-Chair
Professor Marc Snir
Professor Luay Nakhleh, Rice University

ABSTRACT

A critical step in many biological studies is the estimation of evolutionary trees (phylogenies) from genomic data. Of particular interest is the species tree, which illustrates how a set of species evolved from a common ancestor. While species trees were previously estimated from a few regions of the genome (genes), it is now widely recognized that biological processes can cause the evolutionary histories of individual genes to differ from each other and from the species tree. This heterogeneity across the genome is phylogenetic signal that can be leveraged to estimate species evolution with greater accuracy. Hence, species tree estimation is expected to be greatly aided by current large-scale sequencing efforts, including the 5 000 Insect Genomes Project, the 10 000 Plant Genomes Project, the (∼60 000) Vertebrate Genomes Project, and the Earth BioGenome Project, which aims to assemble genomes (or at least genome-scale data) for 1.5 million eukaryotic species in the next ten years. To analyze these forthcoming datasets, species tree estimation methods must scale to thousands of species and tens of thousands of genes; however, many of the current leading methods, which are heuristics for NP-hard optimization problems, can be prohibitively expensive on datasets of this size. In this dissertation, we argue that new methods are needed to enable scalable and statistically rigorous species tree estimation pipelines; we then seek to address this challenge through the introduction of three supertree-like methods: NJMerge, TreeMerge, and FastMulRFS. For these methods, we present theoretical results (worst-case running time analyses and proofs of statistical consistency) as well as empirical results on simulated datasets (and a fungal dataset for FastMulRFS). Overall, these methods enable statistically consistent species tree estimation pipelines that achieve comparable accuracy to the dominant optimization-based approaches while dramatically reducing running time.

ACKNOWLEDGEMENTS

There are many people without whom this dissertation would not have been possible; this section names some of them.

It is fitting that I begin by acknowledging Professors Bill Gropp and Luke Olson. They welcomed me into the scientific computing group, invested their time in training me, and when I wanted to pursue computational biology, they encouraged me to work with Prof. Tandy Warnow. I am grateful for all that they have done for me and for their continued mentorship.

There are no words to describe the gratitude that I feel for being advised by Prof. Tandy Warnow. She has dedicated, what felt like, a limitless amount of time and care into my training, and I have grown tremendously as a result. I am grateful that she introduced me to the field of phylogenomics, a field which I have a grown to love, and I have enjoyed our many discussions on this and related topics throughout the years. I am especially grateful for her continued encouragement and enthusiasm, which is often the inspiration that I need to keep going. She has helped me live a wonderful and opportunity-filled life during graduate school, and I hope that we will continue researching together for many, many years.

My advisors, Profs. Tandy Warnow and Bill Gropp, and my committee members, Profs. Luay Nakhleh and Marc Snir, deserve many thanks for their discussion of and feedback on the work presented in this dissertation. Prof. Marc Snir provided detailed comments on drafts of this dissertation, including the suggestion that I add a list of terminology and acronyms; if you use these lists, which I may or may not have implemented properly, thank him.

I have greatly benefited from the time given to me by many other mentors, collaborators, and co-authors over the years, including Prof. Rasmus Birn, Prof. Rémi Patriat, Prof. Siavash Mirarab, Chao Zhang, Dr. Celine Scornavacca, Prof. Sébastien Roch, Brandon Legried, Prof. Claudia Solís-Lemus, Prof. Mihai Pop, Prof. Todd Treangen, Prof. Mark Springer, Dr. John Gatesy, Prof. Satish Rao, Prof. Vineet Bafna, Prof. Jim Leebans-Mack, Prof. Aron Barbey, Prof. Becky Stumpf, Prof. Gene Robinson, and Prof. Michael Heath. I would be remiss not to acknowledge those, especially my advisors with Profs. Luay Nakhleh, Luke Olson, Rasmus Birn, and Mihai Pop, who encouraged me to pursue an academic career.

The Warnow lab has been a stimulating place to be a graduate student, and I have enjoyed working with many students and post-docs, including Kodi (Collins) Taraska, Nidhi Shah, Xilin Yu, Dr. Nam Nguyen, Thien Le, Jed Chou, Ashu Gupta, Vlad Smirnov, Marc

Canby, Srilakshmi Pattabiraman, Ananya Yammanuru, Qikai Yang, Emma Hamel, Binghui Cheng, Aditi Ghosalkar, Nikhil Agarwal, Dr. Ruth Davidson, and Dr. Paul Zaharias. Sarah Christensen, Dr. Mike Nute, and Dr. Pranjal Vachaspati have been my partners on this journey (a phrase that I have stolen from Mike Nute). I have greatly enjoyed working with them, traveling to conferences with them, and procrastinating with them. I am especially grateful to Mike Nute who was my remote work partner during the time that both of us were away from Champaign-Urbana and who continues to field my "emergency statistics" questions with enthusiasm.

During graduate school, I have been fortunate to participate in the Engineering Graduate Student Advisory Committee (thanks to Rhonda McElroy, Prof. Harry Dankowicz, and Prof. Andreas Cangellaris), the Graduate Committee for the Society of Women Engineers (thanks to the financial support from organizations on and off campus), and the Center for Innovation in Teaching and Learning (thanks to Lucas Anderson). During the fall of 2018, I was a resident at the Institute for Pure and Applied Mathematics (IPAM); this program was an excellent opportunity for developing my research agenda, so I thank IPAM, the organizers, and the participants.

I also recognize the people who have worked hard so that my time as a graduate student could be easier and more productive, including those from the Computer Science Department (Prof. Nancy Amato, Maggie Chappell, Kara MacGregor, Viveka Kudaligama, Samantha Smith, Samantha Hendon, Alayna Timbo, Elaine Wilson, Kathy Runck, and others), the National Center for Supercomputing Applications (Stacy Walker and others), the University of Illinois at Urbana-Champaign Campus Cluster, and the Illinois Data Bank (Hoa Luong and others).

I am grateful for the financial support of Ira and Debra Cohen (through their Graduate Fellowship in Computer Science) and the National Science Foundation (through its Graduate Research Fellowship Program and through Grant No. CCF-1535977 to Tandy Warnow). All computational experiments presented in this dissertation were performed on the Illinois Campus Cluster and the Blue Waters Supercomputer. The Illinois Campus Cluster is a computing resource that is supported by funds from the University of Illinois at Urbana-Champaign and is operated by the Illinois Campus Cluster Program in conjunction with the National Center for Supercomputing Applications. The Blue Waters Supercomputer is supported by the National Science Foundation (Grant Nos. OCI-0725070 and ACI-1238993), the State of Illinois, and as of December 2019, the National Geospatial-Intelligence Agency. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and the National Center for Supercomputing Applications.

To conclude this section, I thank my family; it is through their hard work and good fortune that I have been able to live a life of privilege. My dad, James Molloy, deserves credit for encouraging me to study science and mathematics, and my mom, Katherine Molloy, deserves credit for teaching me how to read (a stressful after school project for both of us). Furthermore, my mom, in reading this entire dissertation, has found many typographical errors; if you are spared typos thank her (along with Profs. Tandy Warnow and Marc Snir), and if you find typos blame me. I also thank the family and friends who have supported me during graduate school, some of whom I name here: Jamie and Albert Molloy Rivero, James (Sr.) and Barbara Molloy, Joan Leister, Susan Swartz, Elizabeth, Jerry, and Shiela Kruepke, Alexis Sheehan, Kristen Vaccaro, Jacqueline McClearly, Margaret Viola, Kathy Breen, Menaka Sampath, Amanda Bienz, Erin Carrier, Imani Palmer, Megan Davis, Kjell Hanson, Chris and Becca Schumann-Cervantes, Kazem and Maggie Alidoost, Tony and Jasmine Navar-Lathers, and Steve, Sharon, and Lynn from Living Yoga Center. If not for Dr. Brian Shah, who reconstructed my jaw after my second year of graduate school (with Drs. Iwei Huang, Madan Kandula, and Thomas Smithy contributing to this multi-year process), it seems unlikely that I would have entered my third year of graduate school. Finally, I thank my partner, Michael Kruepke, and our sweet cat, Alice, for everything.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

An "unlimited thirst for genome sequencing" [1] is transforming research in many domains. Evolutionary genomic biology is no exception, as demonstrated by the 5 000 Insect Genomes Project [2, 3], the 10 000 Plant Genomes Project [4], the (∼60 000) Vertebrate Genomes Project [5], and the Earth BioGenome Project [6], which aims to assemble genomes (or at least *genome-scale* data) for 1.5 million eukaryotic species in the next ten years. These ultra-large datasets will be leveraged to study how species evolve/adapt to their environments and how biodiversity is created/maintained. A crucial step in addressing these and other biological questions is the estimation of evolutionary trees (*phylogenies*).

It is well established that genetic material (DNA) can change over time; for example, one state (representing one of the four nucleotides: Adenine, Cytosine, Guanine, or Thymine) may be substituted for another. DNA evolution is typically modeled as a stationary, reversible, and homogenous (SRH) Markov process, parameterized by a rooted tree with edge lengths indicating the expected number of substitutions (per site). Of these standard models, the simplest, introduced by Jukes and Cantor [7], specifies that all 12 substitutions (e.g., $A \rightarrow C$, $C \rightarrow A$, etc.) occur at equal rates and that all four states occur at the root with equal probabilities. The *Jukes-Cantor (JC)* model is not very realistic, as transitions ($A \leftrightarrow G$ and $C \leftrightarrow T$) are more likely than transversions ($A \leftrightarrow C$, $A \leftrightarrow T$, $C \leftrightarrow G$, and $G \leftrightarrow T$) [8]. The Kimura 2-parameter model [8] allows transitions to occur at different rates than transversions, and the *Generalized Time Reversible (GTR)* model [9] allows all six transitions/transversions to occur at different rates and all four states to occur at the root with different probabilities.

Despite these differences, standard models of DNA evolution define the same generative process: a character state (A, C, G, or T) is drawn from the probability distribution of states at the root and then evolves down the tree, undergoing substitutions. The data observed at the leaves of the tree is referred to as a *site* or site pattern. Repeating this process produces a character matrix, where each column is a site and each row is a leaf. Phylogeny estimation is the reverse; for example, we might seek the model tree (topology and numerical parameters) that maximizes the likelihood of an observed character matrix being generated under a particular model of DNA evolution [10]. *Maximum likelihood (ML)* methods seek model trees with unrooted topologies, as the likelihood of an SRH model tree is independent of the root.

Prior to phylogeny estimation, a character matrix is assembled for a set of species. This requires collecting DNA sequences (from genomes of individuals representing these species) that evolved from the same DNA sequence in the genome of a common ancestor. Although the collected DNA sequences correspond to the same *gene*, because mutations (insertions and deletions) can accumulate over time, they may not have the same length. *Gap* character states ($-$) are added to the DNA sequences so that every two of nucleotides in the same column are *homologous*, meaning they evolved from the same nucleotide in the genome of a common ancestor. The resulting character matrix is referred to as a *multiple sequence alignment (MSA)*. MSAs, like phylogenies, are estimated from data, and this task has its own computational and statistical challenges [11, 12].

Phylogenomics [13] combines phylogeny estimation with *genome-scale* data, meaning that data from across the entire genome (maybe even the whole genome) is available for species of interest. Genome-scale enables the assembly of *multi-locus datasets*, which contain sets (one per gene) of unaligned DNA sequences (typically one per species). After an MSA is estimated for each gene, phylogeny estimation can proceed in the usual fashion by combining the resulting MSAs into one big matrix (referred to as the *concatenated alignment*) and seeking the ML model tree under a standard model of DNA evolution. However, this practice may be inappropriate, as biological processes can cause the evolutionary histories of individual genes (*gene trees*) to differ from each other and from the evolutionary history of the species (*species tree*) [14, 15, 16]. In other words, standard models of DNA evolution assume that all sites evolve down a common tree topology, and this assumption can be violated when the input MSA contains data from multiple genes.

The observation that gene trees can differ from each other and from the species tree combined with the increasing availability of genome-scale data has driven method development in recent years. We follow suit, focusing on species tree estimation in the presence of incomplete lineage sorting and gene duplication and loss.

Incomplete lineage sorting (ILS) [16, 17, 18, 19] is a possible outcome of ancestry. A gene is passed from one individual to another through reproduction, so we can trace the inheritance of the gene backward in time; this population-level process is modeled by the Multi-Species Coalescent (MSC) [18, 20, 21, 22]. Complete lineage sorting occurs when the gene genealogy (gene tree) agrees with the species tree, and ILS occurs when these trees do not agree. The latter is more likely whenever there is a *rapid radiation* (sequence of speciation events close together in time). Many major groups are expected to be impacted by ILS, including birds [23], land plants [24, 25], lizards [26], and placental mammals [27]. Hence, species tree estimation in the presence of ILS is receiving considerable attention [28, 29, 30].

Gene duplication and loss (GDL) [14], as its name suggests, occurs when genes are duplicated in or lost from the genome; this can be modeled in a variety of ways, for example the probabilistic model proposed by Arvestad *et al.* [31]. GDL as well as whole genome duplication is common in fungi [32] and plants [25]; however, most methods for species tree estimation assume that genes evolve without duplications or losses. Therefore, prior to species tree estimation a subset of DNA sequences that evolved without duplications is identified for each *multi-copy* gene (gene that appears multiple times in a genome due to duplication events). This task (referred to as *orthology detection* [33, 34]) is still difficult to do correctly [35, 36, 37], so multi-copy genes are often excluded from species tree estimation (e.g., [24, 25]). Methods that can estimate species trees under models of GDL are of increasing interest, as this would enable phylogenetic signal to be extracted from multi-copy genes while avoiding the challenges of orthology detection.

We are concerned with whether methods are provably *statistically consistent* under models of evolution, where gene trees evolve within a species tree under a gene evolution model (e.g., the MSC model or a GDL model), and then sites evolve down each of the gene trees under a DNA evolution model (e.g., the GTR model). Informally, an estimation method is statistically consistent under a model if the error in the estimated model parameters (for our purposes just the unrooted species tree topology) goes to zero, as the amount of data (random samples) generated under the model and given to the method as input goes to infinity. This is equivalent to method reconstructing the correct unrooted species tree topology with probability converging to one, as the amount of data goes to infinity. If we find some condition for which this does not hold, we say that the method is *statistically inconsistent*; furthermore, we say that the method is *positively misleading* if the method reconstructs the incorrect unrooted species tree topology with probability converging to one as the amount of data goes to infinity.

While statistical consistency is an important theoretical property, it only describes method performance under ideal conditions; thus, we are also concerned with whether methods have good performance in practice. The current standard is to benchmark methods for accuracy as well as robustness to error, model misspecification, and other challenging conditions using simulated datasets.

Finally, we are concerned with whether methods scale to datasets with large numbers of genes and large numbers of species. There are three major classes of phylogeny estimation methods: distance methods, optimization methods, and Bayesian methods. Distance methods (e.g., [38, 39, 40, 41]) are typically the fastest (quadratic storage and quadratic or cubic running times, scaling with the number of species); however, optimization and Bayesian methods are preferred by the phylogenomics community. We focus on optimization methods, which are not as computationally intensive as Bayesian methods (e.g., [42, 43]), but nevertheless can be quite costly. Optimization methods are typically heuristics for NP-hard problems, so many (e.g., [44, 45, 46, 47]) use a combination of hill climbing and randomization to search *tree space* (space of all possible phylogenetic trees, which grows exponentially in the number of species) until some convergence criteria are met. These methods do not have deterministic running times, so worst-case running time analysis cannot be provided. However, such methods will be costly when a large numbers of candidate trees need to be evaluated and/or when the objective function is computationally intensive to evaluate for each candidate tree; both conditions can occur for large datasets. The use of dynamic programming (DP) to solve an optimization problem exactly within a constrained search space has emerged as a powerful technique [48], enabling phylogeny estimation methods that are both statistically consistent and polynomial-time [49]. The worst-case running time of such methods is typically a high degree polynomial (scaling with the number of species and the number of genes), so even DP methods can be computationally intensive on large datasets.

In summary, the field of phylogenetics is characterized by computational challenges (e.g., NP-hard problems and compute-intensive objective functions), statistical challenges (e.g., model misspecification), and big data challenges (e.g., large, heterogeneous, and error-ridden datasets). We explore all of these challenges and make the following contributions.

In Chapter 3, we benchmark five of the dominant species tree estimation methods on simulated datasets with varying levels of ILS, phylogenetic signal per gene, and missing data. For each method, we evaluate species tree accuracy as well as changes in accuracy due to *gene filtering* (the removal of genes from a multi-locus dataset based on some predetermined criteria, for example the percentage of missing data). Our results enable us to reconcile conflicting findings from prior studies [50, 51, 52, 53, 54] and offer recommendations for future studies.

We then turn our attention to scaling the best methods studied in Chapter 3 to larger numbers of species, with the goal of maintaining theoretical performance (statistical consistency) and empirical performance (accuracy). We achieve this through the introduction of disjoint tree merger (DTM) methods: NJMerge (Chapter 4) and its improved version TreeMerge (Chapter 5). Both NJMerge and TreeMerge are designed to operate within a novel divide-and-conquer pipeline that (i) divides species into pairwise disjoint subsets, (ii) estimates a tree on each subset, and then (iii) merges the subset trees using auxiliary information, for example the evolutionary distances estimated between (some but not all) pairs of species. This approach has two advantages: first, it avoids supertree estimation (which is typically formulated as an NP-hard optimization problem [55, 56]), and second, it enables the final tree to obey the topological constraints implied by the subset trees (which should be estimated using the best method possible). We prove that divide-and-conquer pipelines using NJMerge and TreeMerge are statistically consistent under standard models of DNA evolution as well as the MSC model. Finally, we evaluate methods on datasets simulated under the MSC model, finding that our divide-and-conquer pipelines dramatically reduce the running time of the best species tree methods studied in Chapter 3 without sacrificing accuracy.

While many methods have been proven to be statistically consistent under the MSC model, very little is known about the statistical consistency of methods when genes can be duplicated or lost. In a recent study, Legried *et al.* [57] observed that several methods, including MulRF [46, 47], achieved superior accuracy to ASTRAL-multi [58], which, at the time of this study, was the only method proven to be statistically consistent under a model of GDL. This finding motivates Chapter 6, in which we prove that the solution to MulRF's NP-hard optimization problem is a statistically consistent estimator of the species tree under a generic model of GDL, provided that adversarial GDL is prohibited. MulRF is not guaranteed to converge to an optimal solution and has a non-deterministic running time, so we propose a new method, FastMulRFS that operates by performing a reduction on the input data and then using DP to solve MulRF's optimization problem exactly within a constrained search space [59]. This technique enables us to prove that FastMulRFS is statistically consistent and runs in polynomial time. Finally, we evaluate methods on biological datasets as well as datasets simulated under the DLCoal model, which allows both GDL and ILS [60]. Our results show that FastMulRFS achieves comparable accuracy to MulRF while being much faster and also compares favorably to the other methods tested (ASTRAL-multi and DupTree [61]).

We hope these contributions are steps towards a larger goal of developing species tree estimation methods that are both statistically rigorous and practical given the large-scale genome sequencing projects currently underway. We conclude in Chapter 7 with a brief summary and a discussion of open challenges and future work.

CHAPTER 2: BACKGROUND

This chapter contains background material referenced throughout this dissertation. Sections 2.1–2.3 introduce phylogenies as graph-theoretic objects, providing the relevant notation and terminology. Concepts from evolutionary genomic biology are presented in Sections 2.4, and models of evolution are described in Sections 2.5.1–2.5.4. Gene tree estimation methods and species tree estimation methods are discussed in Section 2.6 and 2.7, respectively. To allow readers to skip this chapter, we have indexed acronyms and terminology. The first time a key word appears in a chapter or in a section within Chapter 2 only, it is highlighted in blue, indicating a link to the Appendix. The key word is italicized in addition to being highlighted in blue when it is defined in the text.

2.1 PHYLOGENETIC TREES

A *phylogenetic tree* T is a triplet (t, S, ϕ), where t is a tree (connected acyclic graph), S is a set of labels, and $\phi : L(t) \to S$ assigns each leaf of t to a label in S. We require that every label in S map to at least one leaf of T. A tree T is *singly-labeled* if ϕ is a bijection; otherwise, T is *multi-labeled*. Phylogenetic trees are singly-labeled unless otherwise noted. We do not always make an explicit distinction between a phylogenetic tree T and its graph t; for example, we typically say that "T is a tree on label set S" or "T is a phylogenetic tree," denoting its leaf label set, leaf node set, vertex (node) set, and edge set as $S(T)$, $L(T)$, $V(T)$, and $E(T)$, respectively. The edges that are incident with leaves are referred to as *terminal edges*, and the remaining edges are referred to as *internal edges*. When a phylogenetic tree parameterizes a model of evolution, its edges are assigned weights. The phrase "tree *topology*" simply refers to a phylogenetic tree minus edge weights and any other model parameters.

Phylogenetic trees can be either *unrooted* or *rooted*. In an unrooted phylogenetic tree, t is undirected. Leaves are vertices with degree one, and all other vertices are *internal nodes*. For simplicity, internal nodes are required to have degree three or greater, suppressing all internal nodes with degree two. In a rooted phylogenetic tree, t is directed with edges oriented towards the root: a special vertex with out-degree zero (all other vertices have out-degree one). Leaves are vertices with in-degree zero, and all other vertices are internal nodes. For simplicity, internal nodes are required to have in-degree two or greater, suppressing all internal nodes with in-degree one.

Given a rooted tree, we can identify ancestor-descendant relationships. A vertex v is an ancestor of vertex u (and conversely u is a descendant of v) if there exists a directed path from u to v. We say that v is a common ancestor of $R \subseteq S$ if v is an ancestor of every vertex in R; if, in addition, the path between the root and v is longer than the path between the root and any other common ancestor of R, we say that v is the *most recent common ancestor (MRCA)* of R.

An unrooted tree T can be transformed into a rooted tree by picking a node to be the root and directing edges toward the root; this results in a root vertex with in-degree three or greater. Consequently, it is more common to root T by picking an edge, sub-dividing the edge with a new vertex (the root), and directing edges toward the root; this results in a root vertex with in-degree two. A rooted tree can be transformed into an unrooted tree by ignoring edge directions and the root label and then suppressing the node previously designated as the root if it has degree two.

Several other operations are useful when working with phylogenetic trees. A *contraction* operation corresponds to deleting an edge (u, v) but not its endpoints from T and then identifying vertices u and v. We say that tree T' is a contraction of T if T' can be obtained from T through a sequence of zero or more edge contractions. A *refinement* operation is the reverse of an edge contraction, that is, T is a refinement of T' if and only if T' is a contraction of T. A *polytomy* is any vertex with degree greater than three (and in-degree greater than two if T is rooted). If at least one vertex in T is a polytomy, then we say that T is *unresolved*; otherwise, we say that T is *fully resolved*, as no refinements are possible. A *restriction* operation corresponds to deleting leaves assigned to labels in the set $S \setminus R$ from T and suppressing internal nodes with degree two (and in-degree one if T is rooted). In this case, we say that T is restricted to R and denote the resulting tree $T|_R$.

2.2 COMPARISONS BETWEEN TWO PHYLOGENETIC TREES

We now turn to the issue of comparing two unrooted phylogenetic trees. *Compatibility* is an essential concept, originally described by Estabrook *et al.* [62], and we give the definition from Section 3.2.1 in [12].

Definition 2.1 (Tree Compatibility)**.** Let T and T' be unrooted phylogenetic trees on label sets S and $R \subseteq S$, respectively. We say that T is *compatible* with T' if T' is a contraction of $T|_R$.

Definition 2.2 (Tree Agreement)**.** Let T and T' be unrooted phylogenetic trees on label sets S and $R \subseteq S$, respectively. We say that T' *agrees* with T if T' is isomorphic to $T|_R$. Otherwise, we say that T' *disagrees* with T.

If two trees agree, they are compatible, because every tree is a contraction (or a refinement) of itself. Compatibility can be determined for two unrooted trees in polynomial time using bipartitions.

Definition 2.3 (Bipartition)**.** Let $T = (t, S, \phi)$ be an unrooted phylogenetic tree. The deletion of an edge e but not its endpoints from t produces two rooted subtrees t_A and t_B, splitting S into two sets: $A = \{\phi(l) : l \in L(t_A)\}$ and $B = \{\phi(l) : l \in L(t_B)\}$. Therefore, we say that edge e induces *bipartition* $\pi(e) = A|B$. The set of bipartitions induced by T is denoted $Bip(T) = \{\pi(e) : e \in E(T)\}$. If $|A| = 1$ or $|B| = 1$, we say that $A|B$ is a *trivial bipartition*; otherwise, we say that $A|B$ is a *non-trivial bipartition*.

There exists a bijection between $Bip(T)$ and $E(T)$ provided that T has no internal nodes of degree two (as we require). It easily follows that T' is compatible with T if and only if $Bip(T') \subseteq Bip(T|_R)$; see Section 3.2.1 in [12] for details. Therefore, we can extend the concept of compatibility to bipartitions.

Definition 2.4 (Bipartition Compatibility)**.** Let T be an unrooted phylogenetic tree on label set S, and let $\pi = A'|B'$ be a bipartition on label set $R \subseteq S$. We say that bipartition e is *compatible* with T if there exists a bipartition $A|B \in Bip(T)$ such that $A' \subseteq A$ and $B' \subseteq B$.

If two unrooted trees are not compatible, we may quantify the distance between them. One of the most popular metrics is the *Robinson-Foulds (RF)* distance [63].

Definition 2.5 (Robinson-Foulds Distance)**.** The *RF distance* between two unrooted phylogenetic trees T and T' on the same label set is the minimum number of contraction and refinement operations required to transform T into a tree that is isomorphic to T' or vice versa.

Theorem 2.1 (Robinson and Foulds [63])**.** The RF distance between two unrooted phylogenetic trees T and T' on the same label set is equivalent to the bipartition distance.

$$\begin{aligned} RF(T, T') &= |Bip(T) \triangle Bip(T')| && (2.1)\\ &= |Bip(T) \setminus Bip(T')| + |Bip(T') \setminus Bip(T)| && (2.2) \end{aligned}$$

In a fully resolved, unrooted tree T, there are $2|L(T)| - 3$ edges, of which $|L(T)|$ are terminal edges and $|L(T)| - 3$ are internal edges. Two trees T and T' on the same label set can only differ with respect to their internal edges, which induce non-trivial bipartitions; therefore, $0 \leq |Bip(T) \setminus Bip(T')| \leq |L(T)| - 3$. A similar statement can be made regarding

$|Bip(T') \setminus Bip(T)|$. It follows that the RF distance is a value between zero and $2|L(T)| - 6$.

The RF distance is commonly used to quantify error in *simulation studies*, where data is simulated under a model (parameterized by a phylogenetic tree T^* on S) and then a tree T on S is estimated from the simulated data. When computing $RF(T^*, T)$ via Equation 2.2, the first term corresponds to the number of *false negative (FN)* edges (i.e., edges in the true tree that do not exist in the estimated tree), and the second term corresponds to the number of *false positive (FP)* edges (i.e., edges in the estimated tree that do not exist in the true tree). This analysis assumes a bijection between $Bip(T)$ and $E(T)$, which is the case provided that T has no internal nodes of degree two (as we require). In the context of a simulation study, we typically report the normalized RF distance between the true and estimated tree:

$$\frac{RF(T^*, T)}{2|L(T)| - 6} \tag{2.3}$$

This quantity is referred to as the *RF error rate*. When T^* and T are not fully resolved, it can be useful to report the *normalized symmetric difference*:

$$\frac{RF(T^*, T)}{|E(T^*)| + |E(T)| - 2L(T)} \tag{2.4}$$

where the denominator represents the number of internal edges in T^* and T.

Comparing trees based on their RF distance has advantages and disadvantages; see [64, 65] for discussion. Alternatively, distances between two unrooted trees can be computed using quartets in a fashion similar to Equation 2.1.

Definition 2.6 (Quartet)**.** A *quartet* is an unrooted phylogenetic tree with four leaves. This is the smallest unrooted tree (consider that there are zero non-trivial bipartitions for $|S| = 3$ but three non-trivial bipartition for $|S| = 4$). An unrooted tree T induces a set of quartets, denoted $Q(T)$, obtained by restricting T to all possible subsets of four labels.

Other notable metrics for comparing two unrooted trees include the nearest neighbor interchange distance [66, 67] and the matching distance [64]; also see [65] for more information on comparisons between phylogenetic trees.

2.3 SUPERTREES

Suppose we have a set of phylogenetic trees from different phylogenomic studies, so these trees are on different sets of species and were estimated from different genetic markers. Then, we may wish to combine this phylogenetic information into a single tree on the larger set of species. A major ongoing project with this goal is the Open Tree of Life [68]; their efforts (e.g., [69]) have included the development of new *supertree* methods [55], a class of methods characterized by the following input/output:

- **Input:** Set $\mathcal{T}$ of phylogenetic trees
- **Output:** Tree T on label set $S = \bigcup_{t \in \mathcal{T}} S(t)$

Supertree methods have been widely adopted for estimating species trees under the MSC model (although note that when all trees in $\mathcal{T}$ are on the same label set, this problem is referred to as a consensus tree problem rather than a supertree problem). A less popular but notable application of supertree methods is divide-and-conquer phylogeny estimation; see Section 4.1 for further discussion.

Useful for understanding supertree methods is the concept of *compatibility*, which we previously defined for two trees and now extend to a set $\mathcal{T}$ of trees.

Definition 2.7 (Compatibility Supertree)**.** Let $\mathcal{T}$ be a set of unrooted phylogenetic trees. We say that $\mathcal{T}$ is *compatible* if there exists an unrooted tree T on label set $S = \bigcup_{t \in \mathcal{T}} S(t)$ such that T is compatible with every tree in $\mathcal{T}$. If T is minimally resolved, we say that T is a *compatibility supertree* for $\mathcal{T}$; otherwise, we say that T is a refined compatibility supertree.

Determining whether a compatibility supertree exists for $\mathcal{T}$ is NP-complete for $|\mathcal{T}| > 2$, even when every tree in $\mathcal{T}$ is a quartet [70, 71]. While a compatibility supertree may not exist for $\mathcal{T}$, another possibility is to minimize total distance (or conversely to maximize total support) between the output tree T and the input trees in $\mathcal{T}$. Indeed, many of the leading supertree methods are based on optimization, perhaps the most well-known of which is Matrix Representation with Parsimony [72].

We now define three supertree optimization problems for unrooted phylogenetic trees. The first problem is based on quartets.

Definition 2.8 (Maximum Quartet Support Supertree Problem)**.** Let $\mathcal{T}$ be a set of phylogenetic trees. If a tree T^* on label set $S = \bigcup_{t \in \mathcal{T}} S(t)$ is in the set

$$\arg\max_{T} \sum_{t \in \mathcal{T}} |Q(T) \cap Q(t)| \tag{2.5}$$

then we say that T^* is a *maximum quartet support supertree (MQSS)* for $\mathcal{T}$.

The MQSS problem is NP-hard [73, 74], and well-known heuristics include Quartet Puzzling [75], Quartet Max Cut [76], and *Quartet Fiduccia Mattheyses (QFM)* [77]. When the search space for T is constrained by a set Σ of bipartitions on S (i.e., a solution T must also satisfy $Bip(T) \subseteq \Sigma$), the MQSS problem can be solved in polynomial time [49, 78]. We refer to this problem as the *bipartition-constrained* MQSS problem.

The following supertree optimization problems are based on bipartitions.

Definition 2.9 (Maximum Bipartition Support Supertree Problem)**.** Let $\mathcal{T}$ be a set of phylogenetic trees. If a tree T^* on label set $S = \bigcup_{t \in \mathcal{T}} S(t)$ is in the set

$$\arg\max_{T} \sum_{t \in \mathcal{T}} |Bip(T|_{S(t)}) \cap Bip(t)| \tag{2.6}$$

then we say that T^* is a *maximum bipartition support supertree (MBSS)* for $\mathcal{T}$.

Definition 2.10 (Robonsin-Foulds Supertree Problem)**.** Let $\mathcal{T}$ be a set of phylogenetic trees. If a tree T^* on label set $S = \bigcup_{t \in \mathcal{T}} S(t)$ is in the set

$$\arg\min_{T} \sum_{t \in \mathcal{T}} RF(T|_{S(t)}, t) = \arg\min_{T} \sum_{t \in \mathcal{T}} |Bip(T|_{S(t)}) \triangle Bip(t)| \tag{2.7}$$

then we say that T^* is a *Robinson-Foulds supertree (RFS)* for $\mathcal{T}$.

The RFS problem is NP-hard [79], and MulRF [46, 47] is a well-known heuristic. When T is required to be fully resolved, T is a solution to the RFS problem if and only if T is a solution to the MBSS problem. This is easy to see because

$$\begin{aligned} RF(T|_{S(t)}, t) &= |E(T|_{S(t)})| + |E(t)| - |Bip(T|_{S(t)}) \cap Bip(t)| \qquad (2.8)\\ &= \big(2|S(t)| - 3\big) + |E(t)| - |Bip(T|_{S(t)}) \cap Bip(t)| \qquad (2.9) \end{aligned}$$

implies that a solution T maximizes the third term of Equation 2.9. The bipartition-constrained version of the MBSS problem can be solved in polynomial time [59].

Although not discussed, there are many other (types of) supertree methods; see [55, 56, 80] for an entry into this literature.

2.4 SPECIES TREES AND GENE TREES

In Sections 2.1–2.3, we describe phylogenetic trees as graph-theoretic objects, and from this perspective, there is no difference between a gene tree and a species tree. However, the interpretation of these two types of trees is quite different.

2.4.1 Gene Tree

A *gene tree* represents how a set of DNA sequences evolved from the same DNA sequence through a branching process (note that each DNA sequence is a contiguous region of an individual's genome). Going forward in time, the DNA sequence at the root is passed down as a single hereditary unit from individual to individual until it is inherited by the set of individuals whose DNA sequences are labeling the leaves. This unit of heredity (referred to as a *gene*) can range from a single nucleotide (A, C, G, or T) to several thousand nucleotides. Because mutations can occur as the gene is passed down, there can be different variants of the same gene (referred to as *alleles*). The ancestry of two alleles can be traced backward in time, and when these *lineages* trace back to the same ancestor becoming a single lineage, we say that they have *coalesced*. In summary, each leaf in the gene tree represents an allele, and each internal node represents a *coalescent event*.

A gene, as defined above, is also referred to as a coalescent gene or c-gene [81]. It is worth noting that the term gene can also refer to a contiguous region of the genome that codes a protein (although we do not use this definition). In fact, gene trees are estimated from many different types of genetic markers, including *exons* (coding regions of the genome), *introns* (non-coding regions that lie in between exons), and *ultraconserved elements (UCEs)* [82, 83].

The distinction of genes as units of heredity is important, as genomes (belonging to different individuals) can recombine through various biological mechanisms [84], including sexual reproduction (the genomes of two individuals recombine to form the genome of their offspring). Suppose that a contiguous coding region of a genome has two sections, one inherited from each parent. This region does not constitute a gene, based on our definition. Indeed, our definition of a gene tree implies no *recombination* (note that the ancestry of a DNA sequence impacted by recombination is represented by an ancestral recombination graph [85]). Recombination can also occur in organisms that reproduce asexually; for example, DNA can be exchanged between bacteria [86, 87, 88] or viruses [89]. The transfer of genetic material from a donor to recipient is referred to as *horizontal gene transfer (HGT)*, whereas the transfer of genetic material from parent to offspring is referred to as vertical gene transfer.

2.4.2 Species Tree

To form a gene tree, the ancestry of alleles is traced backward in time to a common ancestor. The pool of potential ancestors for an allele is dictated by the *species tree*. The branches of the species tree represent populations of individuals and thus a population of alleles over time. For organisms that reproduce sexually, branching events indicate a population splitting into two or more populations that are (reproductively) isolated from each other. Internal nodes in the species tree are referred to as *speciation events*, and leaves, which represents a population of individuals, are referred to as species. The precise definition of a species is complicated, and indeed, species delimitation (i.e., determining whether different populations constitute different species) is an active subject of research [90, 91, 92, 93]. We refer the interested reader to [94] for further discussion.

2.4.3 Gene Tree Discordance

Although gene trees evolve within a species tree, this does not imply that these trees will be compatible. When gene trees differ from each other and from the species tree, we say that there is gene tree discordance or *gene tree heterogeneity*. We focus on heterogeneity that results from gene genealogical relationships or from gene duplication and loss; both of these processes are modeled by species trees. It is worth noting that heterogeneity can also result from biological processes, such as HGT, that are modeled by a species network [95, 96].

2.5 MODELS OF EVOLUTION

In this section, we describe several models where gene trees evolve within a species tree; we also provide an overview of standard models of DNA evolution.

2.5.1 Multi-Species Coalescent Model

The evolution of gene trees (gene genealogies) within a species tree is modeled by the *Multi-Species Coalescent (MSC)* [18, 20, 21, 22]. This model is parameterized by (T, Θ), where T is a rooted species tree and Θ is a set of numerical values. The set Θ includes the number of generations of each branch, the *effective population size (EPS)* on each branch (which is constant across all generations on the branch), and the EPS above the root (which is constant across all generations above the root). Note that the EPS is simply the number of alleles in a population; for diploid organisms, this is twice the number of individuals in a population, as each individual has two copies of each chromosome. The MSC model defines a generative process where gene trees evolve *independently and identically distributed (i.i.d.)* within the species tree. For each gene, an allele is sampled for each species at the leaves, and then the lineage for each allele grows backward in time (generations) until all lineages have coalesced, forming a rooted, fully resolved gene tree.

We now describe this process for an MSC model species tree given by the *Newick* string $((A, B) : \tau, C)$. This Newick string indicates a rooted tree on species set $\{A, B, C\}$, where A and B are made siblings and then their least common ancestor (A, B) and C are made siblings; there is one internal edge in this tree with length τ. Note that τ is in *coalescent units* (number of generations divided by EPS). For more information on interpreting Newick strings, see Sections 2.2.1 and 2.3.1 of [12].

For each of the three species, we select an allele (denoted a, b, c) at random, and then the lineage for each allele grows backward in time. There is only one lineage on each branch of the species tree terminal to the leaves, so it is not possible for any of the lineages to coalesce. On the branch above the internal node joining A and B, lineages a and b enter the same population and have the opportunity to coalesce.

Under the *Coalescent* model proposed by Kingman [97], coalescent events follow a Poisson process, and the waiting time until the next coalescent event is a random variable drawn from an exponential distribution with rate parameter

$$\frac{i(i-1)}{2N_e} \tag{2.10}$$

where N_e is the EPS and i is the number of lineages and at the previous coalescent event. At the next event, any pair of lineages can coalesce, with equal probability. This implies a constant and sufficiently large population size, non-overlapping generations, random mating, and no selection; see [98] for discussion.

Because the coalescent process is terminated at the end of the branch, possibly before all lineages coalesce, each branch of the species tree is modeled under the *censored coalescent* [22]. The probability that i lineages coalesce into j lineages after t generations, each with an effective population size N_e, is

$$g_{i,j}(t, N_e) = \sum_{k=j}^{i} e^{-\frac{k(k-1)t}{2N_e}} \frac{(2k-1)(-1)^{k-j}}{j!(k-j)!(j+k-1)} \prod_{m=0}^{k-1} \frac{(j+m)(i-m)}{i+m} \tag{2.11}$$

where $1 \leq j \leq i$ [99, 100]. Using Equation 2.11, we can compute the probability that i lineages enter and j lineages exit a branch of the species tree.

Returning to our example, the probability that lineages a and b coalesce on the branch above the internal node joining A and B equals $g_{2,1} = 1 - e^{-\tau}$. The probability that a and b fail to coalesce equals $g_{2,2} = e^{-\tau}$. If a and b fail to coalesce on this branch, then all three lineages enter the population above the root, at this point any pair of the three lineages coalesce with equal probability. The number of generations above the root goes to infinity, so all lineages entering the root will coalesce into a single lineage. Based on this analysis, it is easy to compute the probability of the three possible rooted, fully resolved gene trees on $\{a, b, c\}$ under the MSC model:

$$P(ab|T, \Theta) = 1 - \frac{2}{3}e^{-\tau} \quad \text{and} \quad P(ac|T, \Theta) = P(bc|T, \Theta) = \frac{1}{3}e^{-\tau} \tag{2.12}$$

where ab indicates that a and b are siblings in the rooted gene tree (note that the leaves of a gene tree are typically relabeled by species in the natural way).

Anomaly zone: The MSC model species tree defines a probability distribution on the space of rooted gene trees. When the most probable rooted gene tree disagrees with the rooted species tree, we say that the species tree is in the *anomaly zone* [101]. Similar statements can be made about unrooted gene trees and unrooted species trees [102]. Notably, there is no anomaly zone for unrooted trees on four species [100] or rooted trees on three species [28, 101]; also see Equation 2.12.

Incomplete lineage sorting: *Incomplete lineage sorting (ILS)* occurs when a gene tree differs from the species tree. As shown in Equation 2.11, ILS is more likely when the length (coalescent units) of the internal branch is short. This makes sense as the lineages entering the branch are less likely to coalesce into a single lineage when the EPS is very large and/or when the number of generations is very small. The level of ILS can be quantified for datasets simulated under the MSC model as the average normalized RF distance (Equation 2.1) between the model species tree and the simulated gene trees; we refer to this value as the *average distance (AD)*.

2.5.2 Probabilistic Model of Gene Duplication and Loss

The probabilistic model proposed by Arvestad *et al.* [31] allows genes to be duplicated and lost but ignores ILS. It is parameterized by (T, Θ), where T is a rooted, fully resolved species tree and Θ is a set of numerical values. The set Θ includes the length of each branch in generations and two additional parameters: the duplication rate λ and the loss rate μ, both in the number of events per allele (in the population) per generation. The *gene duplication and loss (GDL)* model defines a generative process where gene trees evolve *i.i.d.* within the species tree. For each gene, an allele is placed at the root of the species tree, and then the lineage for that allele grows forward in time (generations).

On each branch of the species tree, alleles have the opportunity to be duplicated or lost. A duplication causes the lineage for the affected allele to bifurcate; this bifurcation represents a *duplication event* in the gene tree. In contrast, a *loss event* removes the affected allele from the population. Under the Birth-Death model [103, 104], the waiting time until the next event (either a duplication or a loss) is a random variable drawn from an exponential distribution with rate parameter

$$i \times (\lambda + \mu) \tag{2.13}$$

where i is the number of alleles in the population at the last event. In the next event, any of the i alleles can be impacted (either duplicated or lost) with equal probability. The probability that the next event is a duplication or loss equals $\lambda/(\lambda + \mu)$ or $\mu/(\lambda + \mu)$, respectively.

When a lineage reaches an internal node in the species tree it bifurcates, so that it can descend down both the left and right branches below the internal node; this bifurcation represents a coalescent event in the gene tree. Because each coalescent event in the gene tree corresponds to a speciation event in the species tree, ILS is prohibited under this model. This process continues until all lineages reach the leaves of the species tree (these lineages are labeled according to the leaves of the species tree) or have been lost (these lineages are pruned from the gene tree). The result is a multi-labeled gene tree, typically referred to as a *multi-copy* gene tree, *MUL-tree*, or *gene family* tree.

Paralogs and orthologs: We say that two alleles are *paralogous* if their MRCA represents a duplication event; if their MRCA represents a speciation event, we say that the two alleles are *orthologous*. A set of alleles is orthologous if every pair of alleles in the set are orthologous.

2.5.3 A Unified Model of Duplication, Loss and Coalescence (DLCoal)

The probabilistic GDL model proposed by Arvestad *et al.* [31] ignores population-level effects, such as ILS. Rasmussen and Kellis [60] seek to address this in a unified model of *Duplication, Loss, and Coalescence (DLCoal)*.

Mutations in a population: When a mutation arises at a particular locus, alleles at this locus have two variants: mutant or non-mutant. Initially, the allele frequency of the mutant variant is $1/(2N)$, where N is the number of diploid individuals in the population at the time of the mutation. Changes in allele frequency will occur as the mutant and non-mutant variants are inherited by future generations via random sampling; this is referred to as genetic drift.

Under the *Wright-Fisher (WF)* [98] model, genetic drift is governed by a forward-time

Markov chain with a discrete state space representing the number of alleles with the mutant variant in the population. Let i be the number of alleles with the mutant variant in the current generation. Then, the number of alleles with the mutant variant in the next generation is a random variable drawn from a binomial distribution, where the number of trials is $2N$ and the probability of success is $i/(2N)$. In other words, the next generation of alleles is created by binomial sampling alleles in the current generation with replacement. If over time, all alleles in a population correspond to the mutant variant, we say that the mutant variant has fixed. As the number of generations goes to infinity, the probability that the mutant variant fixes goes to $i/(2N)$ [98]. Therefore, if mutations occur at a particular locus with rate μ (number of events per allele in the population per generation), the probability that the mutation occurs at a particular locus and then the mutant variant fixes is

$$\mu 2N \times \frac{1}{2N} = \mu \tag{2.14}$$

as the number of generations goes to infinity. This implies no new mutations (which alter allele frequency) at the locus, no recombination at the locus, a constant EPS ($2N$ alleles in each generation), non-overlapping generations, random mating, and no selection; see [98] for discussion. Many of the assumptions made by WF model also are made by the Coalescent model. In fact, these two models are related: the Coalescent model can be viewed as an approximation to the WF model for large populations (so that there is at most one coalescent event per generation [105]).

Duplication and loss in a population: Rasmussen and Kellis [60] consider duplications and losses as mutations occurring in a population of diploid individuals. When an allele is lost from the genome of an individual, the loss occurs at a particular locus. Therefore, the population of alleles at this locus now have two variants: either the loss is "present," or it is "absent." When an allele is duplicated in the genome of an individual, a copy of the allele appears at a different locus. Therefore, the mother locus continues to have one variant, but the daughter locus has two variants: either the duplication is "present," or it is "absent." If the mother and daughter loci are unlinked, that is, they evolve independently, then the two loci should be modeled separately with identical parameters (number of generations and EPS) under the WF model (if going forward in time) or the Coalescent model (if going backward in time).

For simplicity, suppose that the duplication-present variant fixes, and then we sample two alleles at random from the population and trace their lineages backward in time.

- If two alleles are sampled from the daughter locus, their lineages are in a common population and may coalesce; in fact, they must coalesce before (more recently than) the duplication event.

- If two alleles are sampled from the mother locus, their lineages are in a common population and may coalesce.
- If one allele is sampled from the mother locus and one allele is sampled from the daughter locus, their lineages are in separate populations until the time of the duplication event, at which point they enter a common population and may coalesce.

To deal with these different scenarios, Rasmussen and Kellis [60] suggest that gene trees evolve within a *locus tree*. This is the basis for the *Multi-Locus Coalescent (MLC)* model [60]. An MLC model locus tree is similar to an MSC model species tree, but internal nodes correspond to either speciation events or duplication events. In the former case, the descending branches are modeled by the censored coalescent; in the latter case, one descending branch (the mother locus) is modeled by the censored coalescent and the other descending branch (the daughter locus) is modeled by the bounded coalescent [60]. The MLC model makes several assumptions, including that duplications and losses (either the present or the absent variants) are fixed at the leaves of the locus tree. Rasmussen and Kellis [60] use this assumption to justify a model (referred to as the DLCoal model) in which gene trees evolve within a species tree in two separate phases.

The DLCoal model is parameterized by (T, Θ), where T is a rooted, fully resolved species tree and Θ is a set of numerical values. The set Θ includes all numerical parameters for the MSC model as well as a duplication rate and loss rate for the GDL model. In the first phase, a locus tree evolves within the species tree under the GDL model proposed by Arvestad *et al.* [31], after which the branches of the locus tree are relabeled based on branches of the species tree in the natural way. In the second phase, a gene tree evolves within the locus tree under the MLC model. This process results in a collection of paired locus trees and gene trees. The locus trees differ from the species tree due to GDL only, a locus tree differs from its gene tree due to ILS only, and the gene trees differ from the species tree due to both GDL and ILS. Therefore, the level of ILS in simulated datasets can be quantified by averaging normalized RF distance (Equation 2.1) between every locus tree and its corresponding gene tree, both with leaves labeled by alleles rather than species (so that the two trees are singly-labeled).

2.5.4 Models of DNA Evolution

In Sections 2.5.1–2.5.3, we discuss models where gene trees evolve within a species tree; however, mutations (e.g., insertions, deletions, and substitutions) can accumulate in genes over time. In fact, mutations are essential for reconstructing gene trees from DNA sequences.

DNA evolution is typically modeled as a Markov process with state space $\mathcal{A} = \{A, C, G, T\}$ and parameters (T, Θ), where T is a rooted gene tree and Θ is a set of numerical values. The set Θ includes the probability distribution $\vec{\pi}$ of states at the root as well as parameters for computing the probability transition matrix P on each branch. The probability transition matrix is P where $P_{i,j}(e)$ is the probability that state i transitions to state j on branch t. In this section, the term transition refers to a substitution event changing state i into state j, where $i, j \in \mathcal{A}$ such that $i \neq j$ (note that this differs from the meaning of the term in Chapter 1).

Such models define a generative process where sites evolve down the gene tree. For each site, a random character state $i \in \mathcal{A}$ is drawn from $\vec{\pi}$ and is placed at the root of the gene tree. This can be achieved by using $\vec{\pi}$ to partition the interval $[0, 1]$ into four sub-intervals (each representing a different state) and then generating a random number between zero and one; the interval in which the random number lands determines the state at the root [106]. This process is repeated at each child of the root except that the character state is drawn from the probability distribution given by the i^{th} row of $P(e)$, which is the probability distribution for a chain in state i transitioning on branch e. Continuing in this fashion, character states are generated at each node in a preorder traversal, producing a site pattern.

No Common Mechanism model: Under the *No Common Mechanism (NCM)* model of Tuffley and Steel [107], each site and edge is allowed to have its own transition probability matrix. This model parameterized by (T, Θ), where T is a rooted phylogenetic tree and Θ is a set of numerical parameters. The set Θ includes the probability distribution of states at the root

$$\vec{\pi} = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \tag{2.15}$$

and the probability transition matrix for each branch and site

$$P_{i,j}(e, s) = \begin{cases} 1 - p(e, s) & \text{if } i = j \\ p(e, s)/3 & \text{if } i \neq j \end{cases} \tag{2.16}$$

where $0 \leq p(e, s) \leq 3/4$ denotes the probability that a substitution occurs on branch e for site s. Note that sites are not identically distributed under the NCM model!

Stationary, Reversible, and Homogenous Markov models: *Stationary, reversible, and homogenous (SRH)* Markov models are parameterized by (T, Θ), where T is a rooted gene tree and Θ is a set of numerical values. The set Θ includes branch lengths (expected number of substitutions per site), the probability distribution $\vec{\pi}$ of states at the root, and the *relative* instantaneous transition rate matrix Q (e.g., $Q_{A,C} = 1/4$ and $Q_{A,T} = 1/2$ means that A transitions to T at twice the rate that A transitions to C). The probability transition matrix for each branch is

$$P(t) = exp(Qt) \tag{2.17}$$

where exp denotes the matrix exponential of Qt and t denotes the length of the branch (expected number of substitutions per site). Lastly, unlike the NCM model, sites evolve $i.i.d.$ down the gene tree.

We now summarize the constraints on Q and $\vec{\pi}$ under the the *Generalized Time Reversible (GTR)* model [9], which as its name suggests is the most general of the SRH Markov models, referring the interested reader to Section 1.5 of [108] for a more thorough discussion. First, as $\vec{\pi}$ is a probability distribution,

$$\sum_{i \in \mathcal{A}} \pi_i = 1 \text{ and } \pi_i \geq 0 \tag{2.18}$$

for all $i \in \mathcal{A}$. Second, as Q is an instantaneous transition rate matrix, its off-diagonal entries are required to be non-negative and its rows are required to sum to one; this gives the constraint:

$$-Q_{i,i} = \sum_{j, \in \mathcal{A}, i \neq j} Q_{i,j} \tag{2.19}$$

Third, for a Markov process to be homogenous, Q must not depend on time. Fourth, for a Markov process to be stationary, $\vec{\pi}$ (the probability distribution of states at the root) also must be the probability distribution of states after time t; this property holds when

$$\pi_i Q_{i,i} = \sum_{j \in \mathcal{A}, j \neq i} \pi_j Q_{j,i} \tag{2.20}$$

because the probability that a chain "exits" state i (left term) equals the probability that a chain "enters" state i (right term). Lastly, for a Markov process to be time reversible, the probability that state i transitions to state j must equal the probability that state j transitions to state i; this property holds when

$$\pi_i Q_{i,j} = \pi_j Q_{j,i} \tag{2.21}$$

for all $i, j \in \mathcal{A}$ such that $i \neq j$.

By Equations 2.19–2.21, Q must have the form

$$Q = \begin{bmatrix} * & a \cdot \pi_C & b \cdot \pi_G & c \cdot \pi_T \\ a \cdot \pi_A & * & d \cdot \pi_G & e \cdot \pi_T \\ b \cdot \pi_A & d \cdot \pi_C & * & f \cdot \pi_T \\ c \cdot \pi_A & e \cdot \pi_C & f \cdot \pi_G & * \end{bmatrix} \tag{2.22}$$

where the diagonal entries are given by Equation 2.19. Recall that the entries of Q define relative instantaneous transition rates and the branch lengths define the expected number of substitutions per site (rather than time). This can be achieved by setting $f = 1$ and scaling Q so that the mean mutation rate (expected number of substitutions per unit time per site)

$$\mu = \sum_{i \in \mathcal{A}} \pi_i \sum_{j \in \mathcal{A}, i \neq j} Q_{i,j} = -\sum_{i \in \mathcal{A}} \pi_i Q_{i,i} \tag{2.23}$$

equals one [108].

Other SRH models, including the *Jukes-Cantor (JC)* model [7], place additional restrictions on Q and/or $\vec{\pi}$ and therefore are sub-models of the GTR model. SRH models are sub-models of the *General Markov (GM)* model [109], which does not require stationarity.

Rate heterogeneity across sites: Under SRH Markov models, all sites evolve at the same rate; however, these models can be extended to allow *rate heterogeneity across sites* simply by scaling the branch lengths of the SRH model tree for each site. For example, a constant c is drawn from a distribution, each branch in the model tree is multiplied by c, and then a single site evolves down the re-scaled GTR model tree; this process is repeated for each site. The most popular of the rate heterogeneity models is the *GTR+GAMMA* model [110, 111], where the scaling factor c is drawn from a continuous gamma distribution with shape parameter α and scale parameter $\beta = \alpha$, so small and large values of α correspond to high and low variation in the rates across sites, respectively. The *GTR+CAT* model [110] is a discretized version of the GTR+GAMMA model; specifically, the gamma distribution is discretized into a fixed number of categories, and sites must evolve within a rate in one of these categories.

Molecular clock: When the *evolutionary distance* (expected number of substitutions per site) between the root and each leaf is the same for all leaves, we say that the SRH model gene tree obeys the *strict molecular clock* [112]. By modifying branch lengths, we can generate sequences under SRH models that violate or relax the molecular clock assumption. Under one of the relaxed molecular clock models proposed by Drummond *et al.* [113], each branch length is modified by a random variable drawn from a log-normal distribution. Alternatively, branches could be modified to allow for systematic differences, for example the evolutionary rate can vary across different species as "smaller-bodied species of vertebrates with faster metabolic rates have higher substitution rates than larger-bodied species" [114].

2.6 GENE TREE ESTIMATION

This section contains material previously published in "Large-scale Species Tree Estimation" [115], which was joint work with T. Warnow.

We now provide a brief overview of methods for estimating gene trees under SRH Markov models of DNA evolution. Such methods take an MSA with n rows (DNA sequences) and L columns (sites) as input and return an unrooted tree with leaves bijectively labeled by the input DNA sequences (rows in the input MSA).

2.6.1 Maximum likelihood Methods

A popular approach is to search for the model tree (topology and numerical parameters) that maximizes the likelihood of observing the input data under a particular model of evolution [10]. *Maximum likelihood (ML)* is statistically consistent under the GTR model even when the MSA has gaps, which are treated as missing data [116] (although this assumes the MSA is error-free). However, finding the ML tree is an NP-hard optimization problem [117], so methods use heuristics to search tree space, computing the log-likelihood of each candidate tree. One of the most popular methods, *RAxML* [45] uses pthreads, vector extensions (SSE3, AVX and AVX2), and other techniques to reduce the amount of time required to compute tree log-likelihood. While such optimizations are critical (especially when there are many unique site patterns in the MSA), they do not impact the number of candidate trees evaluated during the tree search. The number of candidate trees evaluated before the search converges (to a local optimum) is unbounded; therefore, we cannot provide a worst case running time analysis. Several other heuristics for ML tree estimation have been developed, including IQTree [118], *FastTree-2* [119], and PhyML [120].

2.6.2 Distance Methods

We now extend the terminology for phylogenetic trees to *dissimilarity matrices* (symmetric matrices with zeros on the diagonal and non-negative values on the off-diagonal). A *phylogenetic distance matrix* D is a triplet (d, S, θ), where d is a dissimilarity matrix, S is a set of labels, and $\theta : \{1, 2, \ldots, |S|\} \to S$ assigns each row of d to a label in S. We require that every label in S map to at least one row of D. A phylogenetic distance matrix D is *singly-labeled* if θ is a bijection; otherwise, D is *multi-labeled*. Phylogenetic distance matrices are singly-labeled unless otherwise noted. We do not always make an explicit distinction between a phylogenetic distance matrix D and its dissimilarity matrix d; for example, we typically say that "D is a dissimilarity matrix on label set S" or "D is a phylogenetic distance matrix," denoting the entry at the i^{th} row and the j^{th} column as $D[i, j]$. Lastly, we let $S(D)$ denote the label set of D and $D|_R$ denote the dissimilarity matrix created by *restricting* D to the rows/columns with indices in the set $\{\theta(i) \in R : i \in \{1, 2, \ldots, |S|\}$.

Distance methods operate by estimating the evolutionary distance between every pair of sequences in an MSA and then, from the resulting phylogenetic distance matrix D, building a tree with leaves labeled by the set $S(D)$, for example by using the well-known *Neighbor Joining (NJ)* method [121]. Recall that an SRH model tree T has branch lengths indicating the expected number of substitutions on that edge, so the evolutionary distance between sequences i and j equals the path distance between leaves i and j in T.

Definition 2.11 (Additive and Nearly Additive)**.** Let T be a tree on label set S, and let $D = (d, S, \theta)$ be a phylogenetic distance matrix. We say that D is *additive* for T if every entry $D[i, j]$ is the sum of the edge weights on the path between the leaf labeled $\theta(i)$ and the leaf labeled $\theta(j)$ in T. We say that D is *nearly additive* for T if every entry $D[i, j]$ differs from the path distance between leaf labeled $\theta(i)$ and leaf labeled $\theta(j)$ in T by less than half of the shortest internal edge in T.

Theorem 2.2 (Atteson [122])**.** NJ, when applied to a phylogenetic distance matrix that is nearly additive for T, returns T.

Evolutionary distances can be estimated by computing the fraction of sites that differ between the two aligned sequences; however, this does not account for multiple substitutions at a site. Therefore, this value is typically corrected under a model of DNA sequence evolution, for example the JC model. The JC-corrected distance between two aligned sequences converges to the true evolutionary distance [12], as the amount of data generated under the JC model goes to infinity. Therefore, as the amount of data generated under the JC model goes to infinity, D converges to a tree that is additive for the JC model tree, and by Theorem 2.2, NJ returns the JC model tree with probability converging to one. It follows that this approach is statistically consistent under the JC model. A similar result holds for using *log-det distances* to estimate gene trees under the GM model [123].

When evolutionary distances are estimated from finite amounts of data, D may not be additive, and furthermore D may not satisfy the triangle inequality: $D[i,j] \leq D[i,k]+D[k,j]$ for all $i,j,k \in \{1,2,\ldots,n\}$. The latter implies that D is a dissimilarity matrix rather than a distance matrix. Nevertheless, D is commonly referred to as a (phylogenetic) distance matrix.

The distance methods described above run in polynomial time, for example it takes $O(n^2L)$ time to compute JC-corrected distances (or log-det distances) and $O(n^3)$ time to run NJ. Distance methods are typically faster than ML methods, and both types of method are statistically consistent under SRH models of evolution; this begs the question, why are ML methods more popular that distance methods? This is because sequences do not have infinite length, so differences in method performance can be observed in practice; see Chapter 4 for further discussion.

2.6.3 Long Branch Attraction

It is worth thinking about conditions under which methods may fail to recover the correct tree. An important example used in many theoretical and empirical studies is the *Felsenstein Zone* tree [124], a quartet $A,B|C,D$ with a short edge separating A,B from C,D, short edges incident to A and C, and long edges incident to B and D. Recall that edge lengths indicate the expected number of substitutions per site, so this tree violates the strict molecular clock. A single site may undergo multiple substitutions on a long branch, so B and D may be in the same state (even a state which differs from their common ancestor) by chance. When B and D are joined as siblings in the estimated tree, we say that *long branch attraction* has occurred. *Maximum parsimony* (which seeks a tree that minimizes the number of substitutions required to explain the input character data) is positively misleading in the Felsenstein Zone [124]. Although ML and distance methods are statistically consistent, this only describes their performance given infinite sites; indeed, they can fail to recover the correct tree when the number of sites is bounded [125]. Long branch attraction can be prevented by adding more species to the dataset, so that the long branches are broken into several shorter branches; this approach, referred to as taxon sampling, was proposed by Hendy and Penny [126]; see [127] for a review.

2.6.4 Non-parametric Bootstrapping

An important step in phylogenetic and phylogenomic studies is to assign some degree of confidence to each branch in an estimated tree. *Non-parametric bootstrapping*, which was introduced by Felsenstein [128], continues to be one of most popular approaches; also see [129, 130]. At the high level, sites in an MSA are sampled with replacement, a tree is estimated on each bootstrapped MSA, and then confidence in a particular branch (bipartition) is indicated by the percentage of bootstrapped trees that induce the same bipartition. An incorrect branch can have high bootstrap support when there is systematic bias (e.g., due to long branch attraction) or model misspecification. Nevertheless, low *bootstrap support* values (below 50%) are generally considered to be unreliable, whereas high bootstrap support values (above 95%) are generally considered to be reliable [12, 131].

2.7 SPECIES TREE ESTIMATION

This section contains material previously published in "To include or not to include: The Impact of Gene Filtering on Species Tree Estimation Methods" [132] and "Large-scale Species Tree Estimation" [115], both of which were joint work with T. Warnow.

To conclude this chapter, we describe the dominant methods for estimating species trees in the presence of ILS, focusing on worst-case running time as well as statistical consistency under the MSC model (note that we refer to methods that are statistically consistent under the MSC model as *coalescent methods*, although this term does not have an agreed upon meaning). These methods are executed as part of species tree estimation pipelines that begin with the assembly of a *multi-locus dataset*, containing n orthologous DNA sequences (typically one per species) for each of the m genes. An MSA is estimated for each gene, and the resulting MSAs are given as input to the approaches discussed below. Therefore, we are interested in whether methods are statistically consistent under the *MSC+GTR* model, where gene trees evolve within a species tree under the MSC model, and then sites evolve down each of the gene trees under the GTR model (note that there can be differences in assumptions between methods regarding the strict molecular clock, the rate heterogeneity across sites, etc.).

2.7.1 Concatenation Analysis using Maximum Likelihood

In the traditional approach to species tree estimation, MSAs are combined into a single matrix, and then an ML method is used to estimate a tree from the concatenated alignment under a model of DNA sequence evolution. This approach, referred as an unpartitioned *concatenation analysis with maximum likelihood (CA-ML)*, assumes that all sites in the concatenated alignment evolve down the model trees with the same topology (recall that branch lengths can be rescaled for each site under the GTR+GAMMA model). This assumption is violated in the presence of ILS. Simulations under the MSC model have shown that CA-ML can have poor accuracy in the presence of ILS [133], leading to the conjecture, later proven by Roch and Steel [134], that CA-ML can be statistically inconsistent under the MSC model. In fact, Roch and Steel [134] show that CA-ML is positively misleading for a rooted species tree with six leaves in the anomaly zone. There is no anomaly zone for unrooted species trees with four leaves [100], and recently, Wascher and Kubatko [135] showed that CA-ML is statistically consistent under the MSC+GTR model (with some assumptions) for unrooted species trees with four leaves. Notably, the proof assumes that each gene is represented by a single site in the concatenated alignment; see [135] for details. While the conditions under which CA-ML can be relied upon to provide accurate species trees are not fully understood, it continues to be a dominant approach to species tree estimation.

Concatenation analysis using ExaML: It is worth noting that ML methods can run out of memory on large datasets. *ExaML* [136], a distributed-memory version of RAxML, partitions and distributes the concatenated alignment across multiple processors (note that the partitioning must be across sites). Therefore, communication (i.e., the sending/receiving of messages between different processes) is required to compute tree log-likelihood for the *entire* alignment. For example, if the alignment was distributed across p processors, then $\log(p)$ communication steps would be required to compute the log-likelihood for the entire alignment using a standard global reduction; these $\log(p)$ steps are effectively serialized work. Furthermore, the amount of time required for communication is significant compared to other operations, which is why avoiding communication [137], overlapping communication and computation [138], and modeling communication [139, 140, 141] are topics of interest for high performance computing applications. Unless tree space can be navigated effectively, many rounds of communication will be required before a search converges (to a local optimum), so while ExaML is a significant advance in large-scale ML tree estimation, there are open challenges.

2.7.2 Bayesian Co-Estimation Methods

Bayesian *co-estimation* of the species tree and gene trees is considered to be one of the most promising approaches for species tree estimation in the presence of ILS. Examples of co-estimation methods include BEST [142, 143], *BEAST [42], and StarBeast2 [43]. Although simulation studies have demonstrated that these methods can offer substantial improvements in accuracy over other methods [29, 144, 145], one of the most popular co-estimation methods, *BEAST, does not converge in practical amounts of time on datasets with much greater than 25 species and 100 genes [146, 147, 148]. StarBeast2 is an improved version of *BEAST that may scale to somewhat larger datasets.

2.7.3 Gene Tree Summary Methods

A more scalable approach to species tree estimation operates by estimating a gene tree from each MSA and then combining the resulting gene trees into a species tree. The second step is performed using a *gene tree summary method*, for example STAR [149], *STEM* [150], MP-EST [44], NJst [38], iGLASS [151], ASTRAL [49, 152], and ASTRID [39]. We now describe how the dominant summary methods operate and their worst-case running time. All of these methods take unrooted m gene trees, each on n species, as input, unless otherwise noted.

NJst: *NJst* [38] estimates a species tree from the *average gene tree internode distance (AGID)* matrix D, where $D[i, j]$ is the number of internal nodes on the path between leaves i and j, averaged across all gene trees. Technically, the average is taken over the subset of gene trees that contain both i and j, so $D[i, j]$ is undefined when none of the gene trees contain both i and j. If there are no undefined entries in D, a tree is built using NJ. This approach runs in $O(n^2(m + n))$ time and is statistically consistent under the MSC model. The former result is easy to see as the AGID matrix can be computed in $O(mn^2)$ time and NJ runs in $O(n^3)$ time. The latter result follows from a proof by Allman *et al.* [153] showing that the AGID matrix converges to a matrix that is *nearly additive* for the species tree under the MSC model.

ASTRID: *ASTRID* [39] operates in a similar fashion to NJst but runs *FastME* [154] instead of NJ. FastME seeks a tree under the *balanced minimum evolution (BME)* criterion and runs in $O(n^2 \log n)$ time; therefore, the running time of ASTRID is $O(n^2(m + \log n))$. It is not completely clear when NJ or BME is more accurate than the other, although prior studies suggest that BME has a slight advantage over NJ in terms of topological accuracy [39, 155, 156]. FastME cannot run when the dissimilarity matrix has undefined entries; in this case, ASTRID runs BioNJ* [157], a modification of NJ for matrices with missing entries.

ASTRAL: *ASTRAL* [49] estimates a species tree by solving the bipartition-constrained MQSS problem. The solution to this optimization problem is a statistically consistent estimate of the MSC species tree, as there is no anomaly zone for unrooted quartets and as every bipartition in the species tree has a non-zero probability of being induced by one of the gene trees [49]. The latest version of ASTRAL, ASTRAL-III [158], runs in $O(nm|\Sigma|^{1.726})$ time [158, 159], where Σ is the set of bipartitions used to constrain the space of possible solutions. Importantly, Σ contains (at least) all bipartitions induced by the input gene trees, so when the input gene trees are binary and identical to each other, then $|\Sigma| = n - 3$. On the other hand, when the input gene trees are fully resolved and disagree with each other, then $|\Sigma|$ will be much larger. It is possible for each of the gene trees to differ on each of their internal edges, so $|\Sigma| = O(nm)$. Because gene trees can differ from the species tree due to biological processes as well as *gene tree estimation error (GTEE)*, the set Σ will typically be large for many (perhaps most) multi-locus datasets.

MP-EST: Recall that an MSC model species tree defines a probability distribution on gene trees; for example, the probability distribution defined by an MSC model tree with $n = 3$ species is given by Equation 2.11. This allows us to compute the likelihood that an MSC model species tree on $n = 3$ species generated a collection of rooted gene trees (as the frequency of each gene tree topology is an estimate of its probability). This concept can be extended to $n > 3$ species by restricting each of the rooted gene trees to a subset of three species, storing the frequency of each gene tree topology as a vector, and repeating this process for all subsets of three species; this preprocessing step requires $O(n^3m)$ time. The resulting frequency vectors can be used to evaluate the likelihood of a candidate species tree restricted to any subset of three species. By combining the likelihood values computed for all subsets of three species into a single score, we can evaluate the pseudo-likelihood of a candidate species tree. *MP-EST* [44] seeks a model species tree that maximizes this pseudo-likehood function. This search is initiated from a random starting tree, so several independent searches from different random starting trees should be run. Lastly, note that MP-EST requires rooted gene trees as input unlike the other methods discussed in this section.

Theoretical and empirical performance of summary methods: NJst, ASTRID, ASTRAL, MP-EST, and many other summary methods are statistically consistent under the MSC model and have excellent accuracy when given a sufficient number of highly accurate gene trees [38, 39, 44, 49, 152]. Comparisons of summary methods and CA-ML on simulated datasets have suggested that summary methods are more accurate than CA-ML when ILS is sufficiently high and conversely that CA-ML is more accurate than summary methods when ILS is sufficiently low [144, 145, 160, 161, 162, 163, 164].

The current proofs of statistical consistency for many (if not all) summary methods assume error-free gene trees [165]. This is equivalent to assuming that gene trees are estimated in a statistically consistent fashion, which in turn makes the assumption that the number of sites per gene goes to infinity. Recently, Roch *et al.* [125] showed that summary methods can be positively misleading when the number of sites per gene is bounded (for unrooted species trees that generate gene trees in the Felsenstein Zone). This is problematic as recombination-free regions of the genome are expected to be quite short for gene trees with deep coalescent events [81]. Indeed, many simulations have shown that GTEE reduces the accuracy of summary methods [52, 145, 152, 160, 161, 166, 167, 168, 169], suggesting that summary methods may be inappropriate when gene trees cannot be estimated with high accuracy. This raises potential concerns, since low bootstrap support values have been reported for gene trees estimated in several phylogenomic studies [23, 24]. Indeed, our analyses show mean bootstrap support values below 30% for several published multi-locus datasets (Table 3.1), which is suggestive of high GTEE.

Missing data is another common challenge to species tree estimation, as many (or perhaps even most) genes will have some degree of missing data (i.e., missing species) if full genomes are to be utilized; see [170, 171, 172] for an entry into this literature. Nute *et al.* [173] recently showed that ASTRAL and MP-EST are statistically consistent under a model of random missing data, and later Rhodes *et al.* [174] showed that NJst and ASTRID can be statistically inconsistent under this same model. While relatively little is known about the statistical consistency of summary methods under models of missing data, simulations have shown that accuracy can degrade when genes are missing species, especially when the number of genes is limited [39, 172, 175] or when missing data are biased [172].

2.7.4 Site-based Methods

Site-based methods, such as SNAPP [176], SVDquartets [177, 178], and METAL [40] estimate species trees from the concatenated alignment. These methods are characterized by being statistically consistent under the MSC model (plus some model of DNA evolution) when given only one site per gene. Therefore, they are expected to be more accurate than gene tree summary methods when genes have few *parsimony-informative* sites (sites with at least two character states appearing at least twice) and thus high GTEE.

SVDquartets: *SVDquartets* [177, 178], for example, estimates an unrooted species trees on $n = 4$ leaves from site patterns using a technique based on the *singular value decomposition (SVD)*. This approach was recently proven to be statistically consistent under the MSC+GTR model (with some assumptions) for the special case when there is exactly one site per gene [135]. Although one site pattern could be selected uniformly at random from each MSA in a multi-locus dataset, Wascher and Kubatko [135] also provided theoretical justification for running SVDquartets on the concatenated alignment.

For datasets with $n > 4$, SVDquartets operates by restricting the concatenated alignment to subsets of four species and then estimates the species tree using its SVD-based technique, which scales linearly with the number of sites in the concatenated alignment. This produces a set of quartets, which can be assembled into a tree on n species using heuristics for the NP-hard MQSS problem; for example, SVDquartets, as implemented in *PAUP** [179], uses the QFM algorithm. Although quartets can be computed in an embarrassingly parallel fashion, the computation of $O(n^4)$ quartets is still burdensome when n is large. Currently, the SVDquartets pipeline, as implemented in PAUP*, can be run on datasets with large n by randomly sampling a subset of the possible quartets. While this approach has advantages in terms of running time, the accuracy of trees computed using a set of sparsely sampled quartets may be reduced compared to using the full set of quartets [180].

Relatively little is known about the empirical performance (accuracy) of SVDquartets. A recent simulation study by Chou *et al.* [163] found that summary methods (ASTRAL-II and NJst) were more accurate than SVDquartets for long genes and conversely that SVDquartets was more accurate than the summary methods under some conditions with very short genes, likely due to increased GTEE. Notably, SVDquartets was not more accurate than CA-ML under the conditions characterized by very short genes [163]. Because the relative performance of methods can depend greatly on the simulation protocol and model conditions explored, further studies are needed to evaluate the accuracy and running time of SVDquartets.

METAL and related methods: METAL [40] and related methods (e.g., [41]) are distance methods that have been proven to be statistically consistent under the conditions described above. For example, Allman *et al.* [41] showed this approach using log-det distances is statistically consistent under the MSC+GTR model with some assumptions.

CHAPTER 3: SPECIES TREE ESTIMATION IN THE PRESENCE OF INCOMPLETE LINEAGE SORTING

This chapter contains material previously published in "To include or not to include: The Impact of Gene Filtering on Species Tree Estimation Methods" [132], which was joint work with T. Warnow. All supplementary materials referenced in this chapter are freely available on Dryad: doi.org/10.5061/dryad.km24v. Note that plots and tables appear at the end of this chapter in Sections 3.6 and 3.7, respectively.

3.1 INTRODUCTION

Of the dominant approaches for estimating species trees in the presence of incomplete lineage sorting (ILS), gene tree summary methods are growing in popularity, as many are provably statistically consistent under the Multi-Species Coalescent (MSC) model as well as computationally efficient compared to concatenation analysis with maximum likelihood (CA-ML). However, gene tree estimation error (GTEE) and missing data can negatively impact summary methods, and some researchers have concerns about their validity under biologically realistic conditions [81, 181]. As a result, gene filtering based on missing data or proxies for GTEE is an increasingly explored aspect of experimental design. Filtering data, both sites and genes, has a long history in phylogenetics; see [50, 171, 182] for an entry into this literature. Many of these prior studies (e.g., [182, 183, 184, 185, 186, 187, 188, 189, 190]) have focused on CA-ML and/or datasets simulated without gene tree heterogeneity, so their results are not directly applicable to understanding the effect of gene filtering on coalescent methods.

Some recent studies examine this issue on biological datasets (e.g., [50, 51, 52, 53, 54, 169, 171, 191, 192, 193]). Because accuracy cannot be evaluated on biological datasets, different proxies for accuracy were reported, including

- the mean bootstrap support of the estimated species tree,
- the bootstrap support of well-established clades in the estimated species tree,
- the similarity of species trees estimated from the same dataset using different methods, and
- the "stability" of the estimated species tree (comparing the species trees estimated using the same method but given subsets of the genes).

These studies came to contradictory conclusions: some found filtering to be beneficial, while others found filtering to be detrimental. Simulation studies have shown that species tree estimation methods can produce highly supported false positive (FP) branches under some model conditions; see [133] for an example with CA-ML when ILS is high and [162] for an example with summary methods when GTEE is high. Therefore, high bootstrap support or similarity to another estimated species trees may not be reliable indicators of species tree accuracy.

To the best of our knowledge, only three prior simulation studies [191, 192, 194] have directly or indirectly examined the impact of gene filtering on coalescent methods. First, Lanier *et al.* [194] added genes with lower variation when using STEM on datasets with eight species. Second, Liu *et al.* [191] added genes with lower bootstrap support values when using MP-EST on datasets with six species. Third, Huang and Knowles [192] filtered genes with missing data when using the *shallowest divergence* method (a coalescent method that takes site patterns as input) [19] on datasets with eight species. While none of these studies found filtering to be beneficial, many new coalescent methods are now in active use, and the effect of gene filtering likely depends on the method itself as well as on the model condition (e.g., number of species, number of genes, level of ILS, amount of phylogenetic signal across individual genes, etc.) [149]. Hence, a thorough evaluation of gene filtering is needed, especially to examine its effect on some of the current leading species tree estimation methods.

We present the results of a simulation study examining the impact of ISL, phylogenetic signal (across individual genes), missing data, and gene filtering strategies on five species tree estimation methods: ASTRAL-II, ASTRID, MP-EST, SVDquartets, and CA-ML using RAxML. In this study, summary methods (ASTRAL-II, ASTRID, and MP-EST) were more accurate than CA-ML, provided that ILS was sufficiently high and GTEE was sufficiently low. When GTEE was sufficiently high, SVDquartets was more accurate than the summary methods, but otherwise it was one of the least accurate. CA-ML was competitive with (and often outperformed) the other methods under many conditions, including when ILS and GTEE were both extremely high. In general, the relative performance of methods was unaffected by gene filtering based on either GTEE or missing data. SVDquartets and CA-ML did not benefit from either type of filtering, and filtering based on missing data generally reduced accuracy. Filtering genes based on GTEE improved the accuracy of summary methods when the level of ILS was sufficiently low but otherwise reduced accuracy. Exceptions to this trend occurred when the level of GTEE was extremely high, in which case filtering based on GTEE improved summary methods. Importantly, these exceptions were for model conditions with only a few replicates: two replicates with low/moderate ILS, five replicates with high ILS, and 17 replicates with very high ILS.

3.2 PERFORMANCE STUDY

Our study evaluated three summary methods (ASTRAL-II, ASTRID, and MP-EST), one site-based method (SVDquartets), and CA-ML (using RAxML) on a collection of datasets originally simulated by Mirarab *et al.* [152]. We modified these simulated datasets to produce a range of model conditions from relatively easy (e.g., low/moderate ILS, moderate GTEE, and no missing data) to very challenging (e.g., high ILS, high GTEE, and missing data). To explore the impact of gene filtering, genes were removed from datasets prior to species tree estimation based on either GTEE or missing data. Methods were evaluated based on the species tree estimation error, as measured by the RF error rate (Equation 2.3). All software commands are provided in the Supplementary Materials.

3.2.1 Simulated Datasets

In this section, we provide a brief overview of the simulation protocol used by Mirarab *et al.* [152] and describe our modifications to their datasets.

Species trees and gene trees: Mirarab *et al.* [152] used SimPhy [195] to simulate a collection of 200-species, 1 000-gene datasets under the MSC model with six different model conditions: three levels of ILS with either deep or recent speciation. Because MP-EST is computationally intensive on datasets with 50 species [152, 196], we restricted datasets to 26 species (outgroup species and 25 randomly selected species) and used only 20 (out of the original 50) replicates.

After restricting datasets to 26 species, we computed several statistics that reflect the level of ILS: average distance (AD) between the model species tree and the gene trees, the number of model species trees in the anomaly zone (based an empirical estimate), and the number of distinct gene tree topologies. The mean AD ($\pm$ standard deviation) across replicate datasets was $12 \pm 2\%$ for the “low/moderate” ILS condition, $41 \pm 6\%$ for the “high” ILS condition, and $75 \pm 1\%$ for the “very high” ILS condition. Under the low/moderate ILS condition, 20% of the replicate datasets with recent speciation and 60% of the replicate datasets with deep speciation were in the anomaly zone; the number of distinct gene tree topologies was also high: there were 344–442 different topologies (out of 1 000) for replicates with recent speciation events and 509–824 different topologies (out of 1 000) for replicates with deep speciation events. Hence, while AD was only 12%, gene tree heterogeneity in this “low/moderate” ILS condition was still substantial; this condition is representative of species trees with a mixture of short and long edges (in coalescent units), perhaps with a small rapid radiation creating the anomaly zone. In the other model conditions (high and extremely high ILS), 100% of the replicates were in the anomaly zone, and each replicate had 999 or 1 000 distinct gene tree topologies. These high and extremely high levels of ILS are representative of a single clade that has undergone a rapid radiation, so nearly every edge is short. In summary, all model conditions explored have a high incidence of species trees in the anomaly zone but differ with respect to the fraction of branches in the species tree that are short enough for coalescence to be likely.

DNA (gene) sequence data: Mirarab *et al.* [152] used INDELible [197] to simulate DNA sequence data under the GTR+GAMMA model with variable lengths (300–1500 sites) from each model gene tree (with branch lengths deviating from a strict molecular clock). We modified the gene multiple sequence alignments (MSAs) to have shorter lengths (truncating them to the first 100 sites) to produce conditions with fewer parsimony-informative sites and thus higher GTEE. Such conditions may be characteristic of datasets where genes are shortened to avoid recombination (e.g., [198]) or where gene trees have low bootstrap support (e.g., [23, 24, 51, 54]).

Missing data: We modified the gene MSAs (by deleting sequences) to produce conditions with missing data biased towards a random subset of genes. The final datasets had 250 genes missing between 13–19 sequences (i.e., 50–73%), 250 genes missing between 7–12 sequences (i.e., 27–46%), 250 genes missing between three to six sequences (i.e., 12–23%), 150 genes missing two sequences (i.e., 8%), 50 genes missing one sequence (i.e., 4%), and 50 genes missing zero sequences (Supplementary Table S2). The total amount of missing data was approximately 30% for all datasets. Note that our protocol for producing datasets with missing data was based on an empirical dataset published by Hosner *et al.* [51], which shows that the percentage of missing data varies across genes but is uncorrelated with evolutionary rate (Supplementary Table S1 and Supplementary Figure S1).

Estimated gene trees: We estimated maximum likelihood (ML) gene trees under the GTR+GAMMA model using RAxML version 8.2.8 with a single tree search. For each replicate dataset, mean GTEE was computed as the normalized Robinson-Foulds (RF) distance between a true gene tree and its respective estimated gene tree, averaged across all 1 000 genes. Replicates were partitioned based on their mean GTEE as follows. Replicates with the full length sequences (300–1500 sites) were partitioned into *low/moderate* GTEE (i.e., mean GTEE between 0–20%) and *moderate/high* GTEE (i.e., mean GTEE between 20–50%). The mean GTEE averaged ($\pm$ standard deviation) across all replicates with the full length sequences was $16 \pm 2\%$ and $35 \pm 8\%$ for *low/moderate* and *moderate/high* GTEE, respectively (Supplementary Tables S3–S4). Replicates with the truncated sequences (100 sites) had higher GTEE and were partitioned into *very high* GTEE (i.e., mean GTEE within 50–80%) and *extremely high* GTEE (i.e., mean GTEE within 80–100%). The mean GTEE averaged ($\pm$ standard deviation) across datasets with truncated sequences was $69 \pm 8\%$ and $86 \pm 5\%$ for *very high* and *extremely high* GTEE, respectively (Supplementary Tables S5–S6).

3.2.2 Gene Filtering Experiments

We evaluated the impact of gene filtering by removing 0%, 25%, 50%, 75%, 90%, and 95% of the genes, producing a collection of datasets that varied in the number of genes retained for species tree estimation. To filter genes by GTEE, gene trees were sorted from lowest to highest GTEE, and then the top 0%, 25%, 50%, 75%, 90%, and 95% of genes were removed prior to species tree estimation. To filter genes by missing data, gene trees were sorted based on the amount of missing data (i.e., the fraction of species deleted from the gene sequence alignment), and then genes missing greater than or equal to 50%, 25%, 10%, 5%, or 1% of the species were removed prior to species tree estimation. These thresholds for GTEE and missing data resulted in the same number of genes being removed, making the two filtering experiments comparable.

3.2.3 Species Tree Estimation

We evaluated five species tree estimation methods: three summary methods (ASTRAL-II version 4.10.5, ASTRID version 1.1, and MP-EST version 1.5), a site-based method (SVDquartets as implemented in PAUP* version 4a150), and CA-ML (using RAxML version 8.2.8 under the GTR+GAMMA). Of these methods, only ASTRAL-II returns species trees with branch lengths annotated by statistical support; its technique for computing support values has been shown to be a better predictor of topological accuracy than *multi-locus bootstrapping (MLBS)* [199]. ASTRAL-II and ASTRID were given unrooted ML gene trees as input. MP-EST was given ML gene trees rooted at the outgroup species (when available) or at the midpoint of the longest leaf-to-leaf path. The best pseudo-likelihood scoring species tree was taken from ten independent runs of MP-EST.

3.3 RESULTS

We now present the results of four experiments. The first and second evaluate the impact of ILS and phylogenetic signal (across individual genes) on five species tree estimation methods with and without missing data, respectively. The third and fourth evaluate the impact of gene filtering based on GTEE and missing data, respectively.

3.3.1 Impact of Incomplete Lineage Sorting and Phylogenetic Signal

For all methods, mean species tree error increased as ILS and/or phylogenetic signal increased, but the relative performance between methods depended on both ILS and phylogentic signal (Figure 3.1). Phylogenetic signal is reported as mean GTEE, so low phylogenetic signal (per gene) corresponds to high mean GTEE and high phylogenetic signal (per gene) corresponds to low mean GTEE.

For low/moderate ILS (12% AD), CA-ML was the most accurate method for all levels of GTEE (Figure 3.1a). When mean GTEE was less than 50%, all five methods had good accuracy (mean species tree error below 7%). When mean GTEE was between 80–85%, the differences between methods were noteworthy; the mean species tree error was 5% for CA-ML, ranged for summary methods (from 16% for ASTRAL-II to 19% for MP-EST), and was 20% for SVDquartets. Notably, this model condition had only five replicates.

For high ILS (41% AD), the relative performance between methods changed dramatically with GTEE (Figure 3.1b). The three summary methods outperformed SVDquartets and CA-ML when GTEE was low to moderate (i.e., mean GTEE $< 50\%$), but SVDquartets and CA-ML outperformed all summary methods when GTEE was extremely high. CA-ML produced more accurate species trees than SVDquartets except for the highest GTEE condition (a model condition with only four replicates), where both methods had similar accuracy.

For very high ILS (75% AD), results were similar but more pronounced than those observed for the high ILS condition (Figure 3.1c). For lower levels of GTEE, CA-ML and SVDquartets were distinctly worse than the summary methods but still provided reasonable accuracy. All methods decreased in accuracy as the level of GTEE increased, but the accuracy of SVDquartets and CA-ML decreased more gradually than all summary methods. When mean GTEE was at least 90% (a model condition with only four replicates), the differences between methods were dramatic: the mean species tree error for all summary methods was greater than 90%, while the mean species tree error for CA-ML and SVDquartets was much lower at 30% and 37%, respectively.

In summary, both ILS and GTEE had strong effects on the absolute and relative performance of methods. All summary methods typically outperformed or else matched the accuracy of CA-ML when GTEE was sufficiently low, but they were less accurate when GTEE was high. Although SVDquartets was dramatically more accurate than all summary methods under the most difficult conditions (with very high GTEE and very high ILS), SVDquartets was nearly always outperformed by CA-ML. Finally, CA-ML outperformed summary methods for low/moderate ILS and for higher levels of GTEE.

3.3.2 Impact of Missing Data

Missing data nearly always reduced the accuracy of methods (Figure 3.2); however, the reduction in accuracy tended to be fairly small (below 5%). Methods differed somewhat in their response to missing data. ASTRAL-II and ASTRID were quite robust to missing data, with mean species tree error never increasing by more than 6% (and most increases in error were much smaller). MP-EST, SVDquartets, and CA-ML were less robust to missing data, especially for challenging conditions with very high levels of ILS or GTEE. Interestingly, the accuracy of some summary methods *improved* on some datasets with incomplete genes, for example when the level of ILS was very high and the mean GTEE was greater than 85%, a model condition with only nine replicates (Figure 3.2c). Finally, missing data did not impact the relative performance of methods (Supplementary Figure S4).

3.3.3 Impact of Gene Filtering based on Gene Tree Estimation Error

As expected, filtering based on GTEE reduced mean GTEE for the retained genes compared to the original set (Supplementary Figure S8). For example, when GTEE was moderate/high, removing 75% of genes reduced mean GTEE from 35–40% to ∼20% (Supplementary Figure S8a, 8c, and 8e). Despite substantial reductions in mean GTEE, the accuracy of summary methods tended to decrease with filtering except when ILS was low/moderate (Figure 3.3).

For low/moderate ILS (12% AD) and moderate to very high GTEE, filtering improved the accuracy of summary methods, provided that the number of retained genes was not too small (Figure 3.3a and 3.3b). Specifically, removing 75% of the genes based on GTEE reduced mean species tree error by ∼2–3% for the three summary methods. Importantly, CA-ML was the most accurate species tree method for this condition.

For higher ILS conditions, filtering was at best neutral and typically increased species tree error (Figure 3.3c, 3.3d, 3.3e, and 3.3f). Exceptions occurred when GTEE was extremely high (mean GTEE >85%), in which case filtering improved the accuracy of summary methods for all ILS levels (Table 3.2 and Supplementary Tables S10–S11). Notably, the number of replicates with extremely high GTEE was limited: two replicates with low/moderate ILS, five replicates with high ILS, and 17 replicates with very high ILS.

Finally, filtering had minimal impact on ASTRAL-II's local branch support but typically decreased the mean support of the true branches recovered by ASTRAL-II and thus increased the number of true branches with support less than 75% (Supplementary Figures S6–S7). It is also worth noting that, as expected, CA-ML and SVDquartets both decreased in accuracy when the sequence alignments corresponding to genes with high GTEE were removed (Supplementary Figure S5 and Supplementary Tables S12–S13).

3.3.4 Impact of Gene Filtering based on Missing Data

Filtering genes based on missing data typically reduced the accuracy of all methods, but the amount depended on ILS and GTEE as well as the on number of genes retained after filtering (Figure 3.4, Supplementary Tables S14–S18). For all methods under all conditions examined, removing 50% of genes (retaining 500 of the 1 000 genes) had a negligible impact on accuracy; however, as more genes were deleted, the mean species tree error increased. In the most extreme case, 95% of the genes (i.e., all genes with missing data) were removed, so that only 50 of the original 1 000 genes were retained. For the easiest condition (i.e., low/moderate ILS and moderate GTEE), filtering 95% of the genes increased mean species tree error by approximately 5% (Figure 3.4a); thus, the impact of filtering was slightly negative. For the most challenging condition (i.e., very high ILS and very high GTEE), filtering 95% of the genes increased mean species tree error by ∼25% for all methods (Figure 3.4f); thus, the impact of filtering was very negative. Finally, filtering decreased the mean branch support of the true branches recovered by ASTRAL-II, resulting in a greater number of true branches with support less than 75% (Supplementary Figures S9–S10).

3.4 DISCUSSION

The datasets used in our study covered a broad range of model conditions, and as nearly all replicates were in the anomaly zone, species tree estimation with coalescent methods was relevant. However, our study was constrained to five methods and to datasets with 26 species and 1 000 genes (unless gene filtering was performed), and trends may not generalize to other methods or to other datasets with much smaller or much larger numbers of species and/or genes.

3.4.1 Impact of Incomplete Lineage Sorting and Phylogenetic Signal

In our study, both ILS and GTEE impacted the relative accuracy of species tree estimation methods. CA-ML had the best accuracy of all methods under the low/moderate ILS condition, even though many of the replicates were in the anomaly zone. Interestingly, CA-ML was more accurate than coalescent methods for some conditions with very high ILS, specifically when mean GTEE was greater than 85% (a model condition with only nine replicates). However, except when GTEE was sufficiently high, differences in accuracy between CA-ML and the summary methods were usually small. Summary methods were typically more accurate than CA-ML when the level of ILS was not too low and GTEE was not too high (mean $< 50\%$). Thus, summary methods performed close to best (and sometimes best) under many conditions but always provided that GTEE was not too high.

Our observed trends with respect to the relative accuracy of ASTRAL-II, ASTRID, MP-EST, and CA-ML are similar to those from prior simulation studies [39, 49, 161, 162, 163, 164, 200], and they are also consistent with earlier simulation studies evaluating other coalescent methods [38, 144, 145, 160, 201]. The improvement of CA-ML over summary methods has been noted before for high levels of ILS (e.g., [152]) but not (to our knowledge) for conditions with very high ILS (75% AD), as was observed in our study. Finally, the good performance of ASTRID in our study is consistent with prior simulation studies evaluating ASTRID and/or NJst [39, 201].

In our study, SVDquartets was not among the best methods. CA-ML was at least as accurate and usually more accurate that SVDquartets, even when the level of ILS was very high, a condition for which SVDquartets would be expected to have an advantage. Our observed trends agree with that of a prior simulation study [163] by Chou *et al.* evaluating SVDquartets in comparison to other species tree estimation methods; they also found that SVDquartets was less accurate than CA-ML and was typically less accurate than the summary methods examined.

3.4.2 Impact of Missing Data

In our study, missing data slightly reduced the accuracy of methods, a trend that has also been noted in prior studies [39, 172, 175]. Although missing data, in a few cases, improved the accuracy of summary methods, this trend may be explained by decreased GTEE. For the nine datasets with very high ILS (75% AD) and extremely high GTEE (>85%), on average missing data improved the accuracy of 835/950 gene trees; the average reduction in error was 3.6%. One would expect many of these gene trees to have short branches (because species trees with very high ILS have very short branches), so the random deletion of species may have increased the length of some branches, making them easier to estimate.

3.4.3 Impact of Gene Filtering based on Gene Tree Estimation Error

In our study, filtering based on GTEE typically improved the accuracy of summary methods on datasets with low/moderate ILS and typically reduced accuracy of summary methods for datasets with higher levels of ILS. Note that regardless of the ILS level, mean GTEE of the retained genes after filtering was substantially reduced compared to the original set (before filtering). Hence, even when filtering improved the overall quality of the input gene trees, this did not always offset the negative impact of reducing the total amount of genes via filtering.

Differences due to ILS level could be explained as follows. When ILS is sufficiently low, a few highly accurate gene trees are sufficient to estimate the true species tree (e.g., one perfectly estimated gene tree is identical to the species tree in the no-ILS condition). However, a large sample of gene trees is necessary to accurately estimate the species tree under higher levels of ILS. Hence, filtering genes will be detrimental unless a sufficiently large number of genes is retained after filtering, and this number of genes will vary with the level of ILS. This explanation is consistent with a recent mathematical result showing that the number of true gene trees required for ASTRAL to recover the true species tree with high probability grows in proportion to the shortest branch in the true species tree [202] (and recall that ILS is more likely when branches in the species tree are short). Based on this explanation, one would expect gene filtering to have a particularly negative effect under high ILS conditions. Even so, when the level of ILS was high or very high and when the level of GTEE was extremely high (mean $> 80\%$), filtering based on GTEE could improve summary methods. This model condition had very few replicate datasets (only five replicates with high ILS and 17 replicates with very high ILS), so further investigation is warranted.

Filtering based on GTEE affected CA-ML and SVDquartets quite differently; filtering decreased the accuracy of CA-ML and SVDquartets, even when GTEE was very high. Equivalently, CA-ML and SVDquartets benefited from the additional genes, even when the added genes had very low signal. In bypassing gene tree estimation, CA-ML and SVDquartets were more robust to variation in the quality of each gene and improved with additional data regardless of the amount of phylogenetic signal per gene.

To the best of our knowledge, only two prior simulation studies [191, 194] have examined how filtering genes based on GTEE or proxies of GTEE affects coalescent methods. Liu *et al.* [191] performed a simulation on 6-species model trees with high ILS (50% AD, L. Liu, personal communication with T. Warnow) in which there were two types of genes: "strong genes" (which had 1 000 sites) and "weak genes" (which had 100 sites). ML trees estimated on the weak genes had mean bootstrap support less than 40%, while ML trees estimated on the strong genes had mean bootstrap support greater than 80%, suggesting that there was low GTEE for the strong genes and moderate/high GTEE for the weak genes. Species trees were inferred from estimated gene trees using MP-EST. They [191] observed that adding 60 weak genes (in increments of 10) to a set of 30 strong genes increased the fraction of replicates in which the true species tree was recovered from 33% to 50%; however, the improvement was not monotonic (i.e., as weak genes were added, the accuracy of MP-EST sometimes decreased). Based on the ILS level and the number of genes, we would expect that accuracy would improve by including weak genes, so the results in [191] are consistent with our study. Lanier *et al.* [194] performed a simulation study on 8-species model trees with two levels of ILS. Although they [194] found that adding up to 50 low-variation genes to a single variable gene had little impact on STEM; however, each gene was represented by a *strict majority consensus (SMC)* tree (all bipartitions in the SMC tree are induced by greater than 50% of the trees sampled during the MCMC analysis) from MrBayes [203] that may not have been fully resolved due to insufficient phylogenetic signal. Our study used fully resolved ML gene trees, and so it is difficult to compare our results to those of Lanier *et al.* [194].

3.4.4 Impact of Gene Filtering based on Missing Data

In our study, filtering based on missing data did not decrease mean species tree error of any method under any model condition. Low amounts of filtering generally did not impact method accuracy, but large amounts of filtering resulted in increased species tree error for all methods. Unlike filtering based on GTEE, filtering based on missing data did not substantially lower the average GTEE in the retained genes for most model conditions.

To the best of our knowledge, only one prior simulation study [192] has explicitly examined how gene filtering based on missing data affects coalescent methods. Huang and Knowles [192] simulated 8-species datasets using a protocol where gene trees differ from the species tree due to ILS and where the pattern of missing data was similar to those generated by RADtag protocols [204]. This simulation design resulted in a correlation between the genes with missing data and the genes with higher rates of evolution (and thus phylogenetic signal). Huang and Knowles [192] found that filtering genes based on missing data reduced the accuracy of the shallowest divergence method. Gene filtering (which reduces the total number of genes) can be especially detrimental if it also reduces the average phylogenetic signal per gene, so this finding is consistent with our study.

3.4.5 Data Quality versus Data Quantity

Filtering reduces the amount of data available, which should in turn reduce species tree accuracy. However, sometimes filtering based on GTEE improved species tree estimation using summary methods. An examination of the conditions when filtering improved the accuracy of the summary methods shows that the average gene tree accuracy also improved substantially without removing too many genes. Hence, although there was a reduction in data quantity (number of genes), there was an increase in data quality (accuracy of gene trees). Similarly, when filtering reduced accuracy, either the gene tree quality did not improve by filtering (e.g., when filtering is based on missing data) or the gene tree quality improved but not by enough to offset the reduction in quantity. In other words, the impact of filtering is fundamentally a question of data quality versus data quantity.

3.4.6 Branch Support

In our study, gene filtering impacted the branch support values (computed by ASTRAL-II); however, these results are somewhat difficult to interpret as the branches recovered in the estimated species trees were also impacted by gene filtering. Even though gene filtering did impact the support of true and false positive branches in the estimated species trees, branch support was still useful for distinguishing between true and false positives (Figure S6, S7, S9, and S10).

3.4.7 Prior Empirical Studies

Several recent studies evaluated the impact of filtering on coalescent methods using empirical datasets. Four studies evaluated filtering based on missing data [50, 51, 53, 171], and six studies evaluated filtering based on proxies for GTEE or related criteria [50, 51, 52, 53, 54, 193]. These studies differ in many ways, including the dataset type, the filtering criteria, the species tree estimation methods used, and the evaluation of species tree quality; however, all these studies used MLBS to estimate branch support for the species trees computed using summary methods.

Two empirical studies [50, 51] observed that deleting genes based on the degree of missing data typically did not improve the quality of estimated species trees, and sometimes even reduced quality, as measured by the appearance of unlikely clades. The other two studies [53, 171] observed that deleting genes with missing data could either increase or decrease the branch support of likely clades, depending on the threshold that was used to filter genes based on missing data. The best filtering threshold differed between the studies, indicating that the optimal filtering threshold based on missing data may depend on the species tree estimation method and the data set properties in ways that are difficult to ascertain. Given the lack of evidence that filtering in this way improves accuracy [50, 51], filtering genes with missing data seems to be undesirable. An interesting counterpart is the observation by Hosner *et al.* [51] that filtering genes that have fragmentary sequences (a different kind of missing data situation that they refer to as "type-II" missing data) can improve accuracy. The explanation is likely that type-II missing data increased GTEE, which in turn impacted the summary methods.

Five empirical studies have examined how species trees computed using summary methods are impacted by gene filtering based on proxies for GTEE; three of these studies [51, 52, 53] recommended filtering and two of these studies [50, 54] did not recommend filtering. The sixth study [193] explored a related filtering strategy that identified and removed outlier genes (i.e., genes whose gene trees are topologically very distant from other gene trees) but they did not evaluate this type of filtering as a way to improve species tree estimation. The five studies that used proxies for GTEE to evaluate species tree quality did so in similar ways (using the amount of similarity to a CA-ML tree, the appearance of unlikely clades, or the branch support of the estimated species tree), and yet these studies came to different conclusions.

It is worth examining in some detail the study that showed the largest favorable impact of filtering [52] as the authors provided convincing evidence that filtering reduced the appearance of unlikely clades in trees computed by MP-EST. Meiklejohn *et al.* [52] noted that many of the gene trees in their dataset (especially ones with few parsimony-informative sites) had highly supported GTEE. Their evidence for this assertion is that these gene trees had strong support for clades that conflicted with a well-established clade in the species tree and that this well-established clade was separated from the rest of the species by a long branch. Since ILS is unlikely to occur on long branches, this means that most true gene trees would be expected to display this well-established clade. Meiklejohn *et al.* [52] noted that removing genes with few parsimony-informative sites simultaneously removed many of the genes that supported unlikely clades in the species tree.

Chen *et al.* [50] make an interesting point about gene selection strategies that is relevant to the question of gene filtering. They argue that node-specific strategies [184] should be used to select genes, rather than "nonspecific" strategies that select genes based on overall phylogenetic signal and/or assessment of gene tree accuracy. They showed that the selection of genes to help resolve specific phylogenetic questions (e.g., Is Amborella the sister of land plants?) is more likely to result in an answer with high support than selecting genes based on generic signal. For example, Chen *et al.* [50] wrote, “In some extreme cases, these nonspecific datasets can correctly resolve some difficult nodes but result in high support for erroneous relationships for other nodes.” They concluded that “One possible explanation for this phenomenon is that each gene has a different resolving power on different time scales and on different evolutionary scenarios... Nonspecific datasets may produce a well-resolved relationship for an ancient divergence event but do not have enough phylogenetic signal to recover accurate phylogeny for a recent radiation or vice versa.” The observations by Chen *et al.* [50] are very similar to earlier observations made by Townsend and Leuenberger [205], who noted that “characters that are highly informative early in history rapidly become sources of phylogenetic noise due to multiple hits for deeper divergences.” Both of these studies support the hypothesis that species tree estimation is likely to benefit from selecting a mixture of genes aiming to resolve different parts of the tree, rather than selecting genes on the basis of overall high signal.

Even though the empirical studies did not come to any clear agreement regarding the impact of filtering, several conclusions can be drawn. First, not all methods responded identically, so improvements resulting from filtering for one method do not imply that other methods (even of the same type) will respond in the same way. Second, when there was improvement, it was often because the genes that were deleted had high GTEE, with potentially the biggest improvement occurring when the GTEE was highly supported. Third, interpreting increased similarity to a CA-ML tree as an improvement in accuracy is a bit tricky, as it depends on the accuracy of CA-ML. However, our study showed that CA-ML was frequently the most accurate of the species trees estimation methods, even under conditions with localized regions of high ILS (i.e., the species trees have a mixture of long and short branches so that the overall the AD value was not too high). We also showed that filtering genes based on GTEE reduced species tree error when the level of ILS was low/moderate or when the level of GTEE was extremely high. These two conditions seem likely to be true of many empirical studies, and high GTEE, in particular, seems to be a problem for modern multi-locus datasets (Table 3.1).

3.5 CONCLUSIONS

Perhaps the most significant outcome of our study is the observation that GTEE (resulting from low phylogenetic signal per gene) has a substantial negative impact on summary methods. This impact can be so great that summary methods may not be appropriate for analyzing exons and other types of multi-locus datasets where genes have very few parsimony-informative sites. We recommend that prior to data collection researchers consider whether the expected level of ILS is high enough that a summary method would be beneficial and then select markers based on this assessment.

Post-data collection, we recommend that methods be chosen based on the empirical properties of the dataset, considering data quality (per gene) and data quantity (number of genes). Given a large collection of genes, many of which have only a few parsimony-informative sites, we may prefer CA-ML over summary methods, as CA-ML is likely to be at least as accurate as the summary methods, especially if the level of ILS is low/moderate. Given a large collection of genes, many of which appear to produce high quality gene trees, we may prefer summary methods over CA-ML, as summary methods are likely to be at least accurate as CA-ML and more accurate if the level of ILS is high. The conditions for which SVDquartets may be more accurate than summary methods and/or CA-ML require further exploration.

Our study demonstrates that method selection as well as the usefulness of gene filtering for summary methods depends on ILS and GTEE, emphasizing the need to accurately predict ILS and GTEE for empirical datasets. One way to estimate ILS is to examine the species tree for rapid radiations, but this may depend on having a very good estimate of the species tree, which is not always available. Another way is to examine the heterogeneity of estimated gene trees; however, it can be difficult to distinguish between heterogeneity due to GTEE and heterogeneity due to ILS. Hence, estimating ILS generally also depends on our ability to evaluate error in estimated gene trees.

Closely related is the problem of estimating the probability that a branch (in an estimated gene tree) is correct [206]. Non-parametric bootstrapping, one of the most popular techniques, can be reliable when performed correctly and when the estimation method is statistically consistent [129, 130, 207, 208]. Other mathematical and statistical approaches have also been developed [185, 207, 209, 210, 211], and they may provide better indicators of GTEE than non-parametric bootstrapping approach.

Empirical arguments also can be made regarding GTEE. For example, Meiklejohn *et al.* [52] argued for GTEE in their dataset, because a large number of the estimated gene trees had strongly supported branches that conflicted with an established clade in the species tree that was separated by a long branch from the remainder of the tree. Evidence for GTEE also can be provided by demonstrating model misspecification, showing that the gene tree topology is not stable, or identifying (outlier) gene trees that are topologically very distant from the other gene trees. Simmons *et al.* [193] found the latter approach useful for identifying other types of errors in multi-locus datasets that produce poor quality gene trees.

Many multi-locus data sets are based on exons and other highly conserved markers for which individual genes may have low numbers of parsimony-informative sites and thus high

GTEE. As the current coalescent methods do not exceed the accuracy of CA-ML on such datasets, many key evolutionary questions remain unanswered. The development of coalescent methods that can provide high accuracy under such conditions is an important direction for future research.

Consideration should also be given to the computational cost of methods. The concatenation analyses in the Avian Phylogenomics Project [23], which ran ExaML on 48 species and ∼14 000 genes, took more than 250 CPU years and one TB of memory (A. Stamatakis, personal communication with T. Warnow). In contrast, MP-EST took only five CPU years to analyze the same dataset, and most of that time was spent estimating ML gene trees with non-parametric bootstrapping. SVDquartets and other site-based methods also are efficient for large numbers of genes and improved versions of these methods may scale to very large numbers of species.

Fortunately, species tree estimation is a fast moving field with many new theoretical results established and methods created in just the last few years (e.g., [202, 212, 213, 214, 215, 216, 217, 218, 219]). With the increased attention that species tree estimation is receiving, we are optimistic that, over the next few years, new methods may be developed that will provide even better accuracy and scalability to large datasets.

3.6 PLOTS

This section contains the four plots presented in Section 3.3 Results.

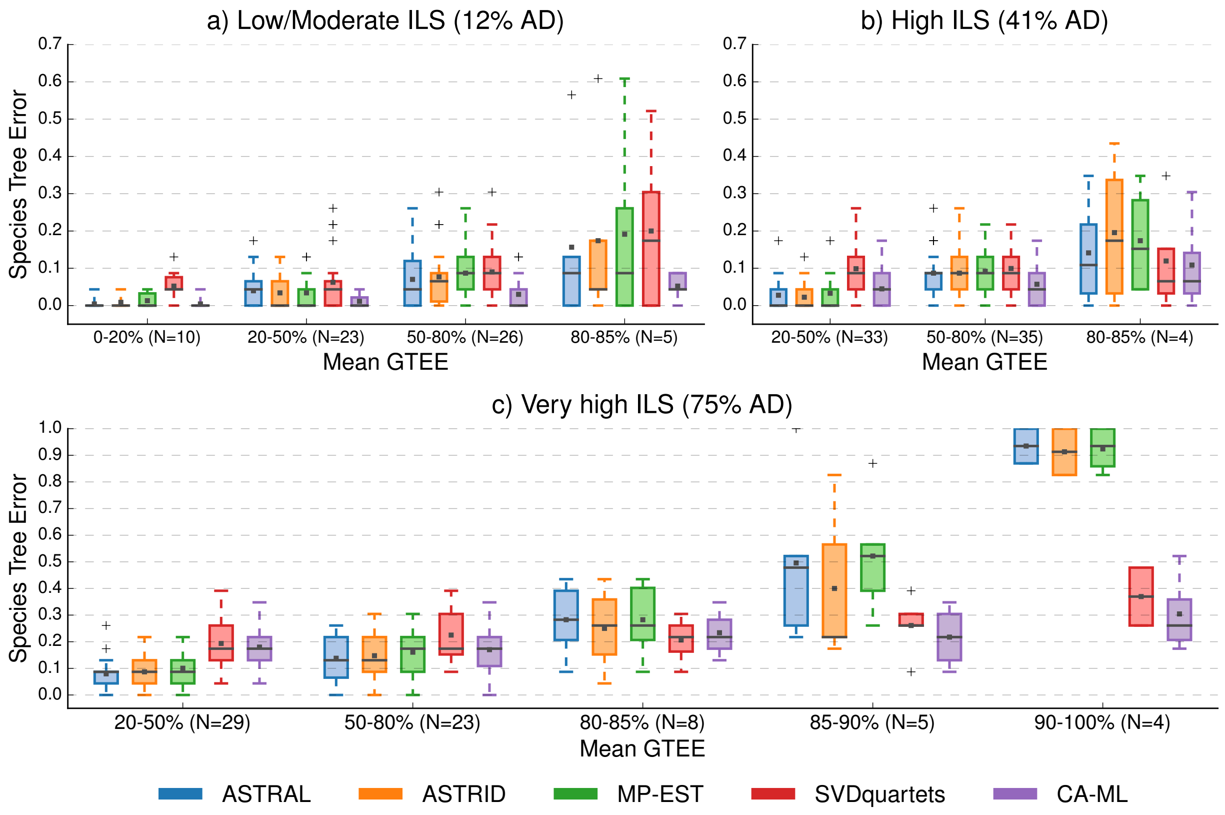


Figure 3.1: **Impact of GTEE.** The impact of GTEE and ILS on species tree error (RF error rate) is shown for methods: ASTRAL-II (blue), ASTRID (orange), MP-EST (green), SVDquartets (red), and CA-ML using RAxML (purple). Subplots a, b, and c show three levels of increasing ILS. The mean GTEE range and the number of replicates (N) for that model condition are given on the x-axis. Means and medians are denoted by the gray dot and bar, respectively. Box plots are defined by quartiles, e.g., boxes extend from the first to the third quartiles. Greater levels of ILS and/or GTEE increased species tree error rates for all methods, and the relative performance of methods depended on both ILS and GTEE. Under low to moderate ILS, CA-ML tended to have better accuracy than the coalescent methods. Under higher levels of ILS, summary methods were typically more accurate than CA-ML and SVDquartets except for conditions with high GTEE.

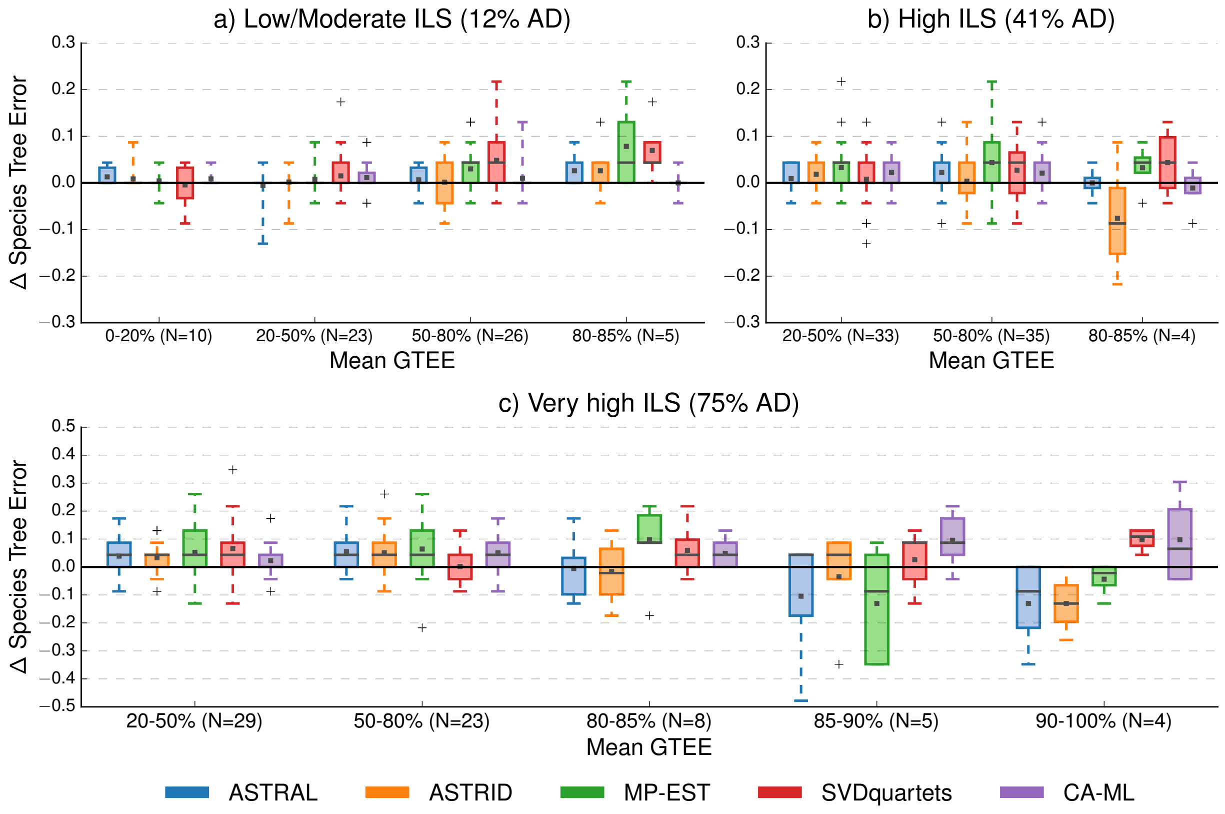


Figure 3.2: **Impact of missing data.** Differences in species tree error between datasets with no missing data and datasets with approximately 30% missing data are shown for five methods: ASTRAL-II (blue), ASTRID (orange), MP-EST (green), SVDquartets (red), and CA-ML using RAxML (purple). Positive values indicate increases in error, whereas negative values indicate reductions in error. Subplots a, b, and c show three levels of increasing ILS. The mean GTEE range and the number of replicates (N) for that model condition are given on the x-axis. Means and medians are denoted by the gray dot and bar, respectively. Box plots are defined by quartiles, e.g., boxes extend from the first to the third quartiles.

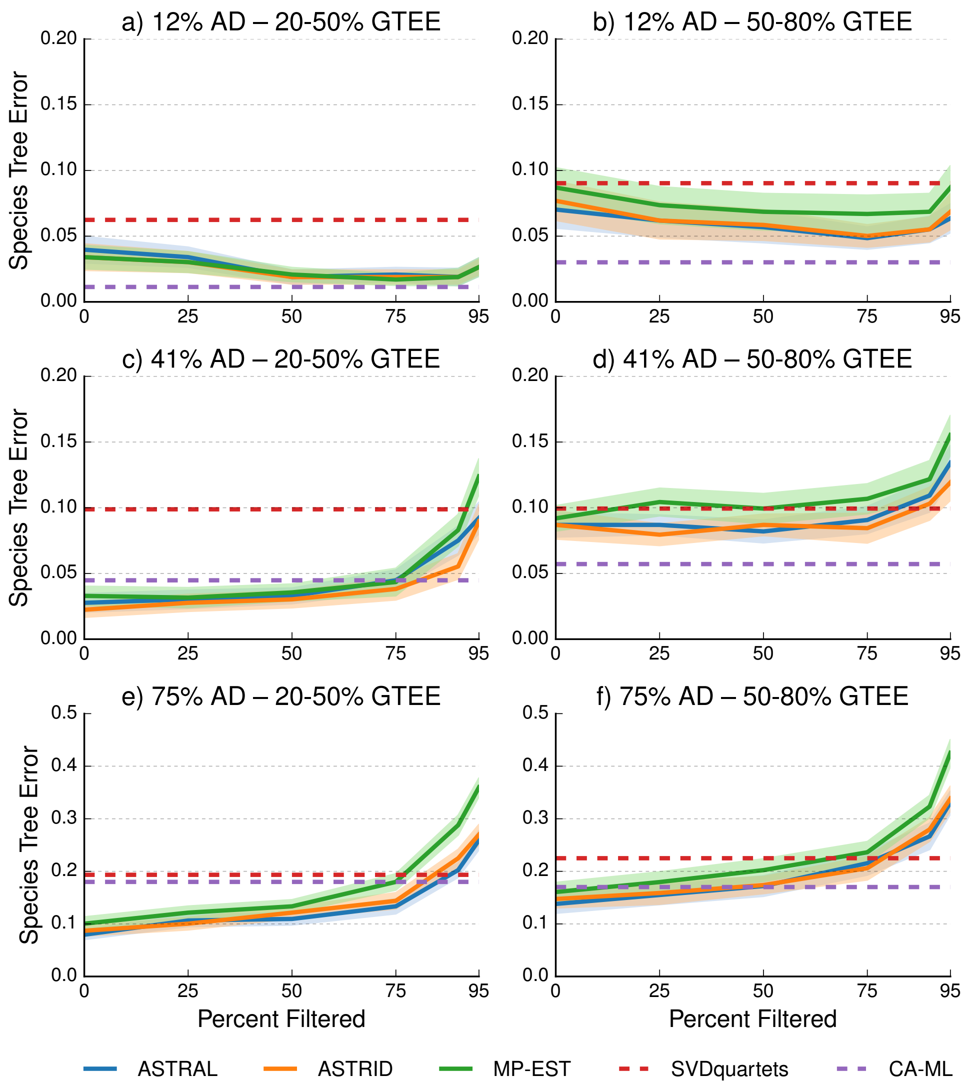


Figure 3.3: **Impact of filtering based on GTEE.** The impact of filtering genes based on GTEE on species tree error (RF error rate) is shown for three summary methods: ASTRAL-II (solid blue), ASTRID (solid orange), and MP-EST (solid green). Genes were filtered by removing the top 25%, 50%, 75%, 90%, and 95% of genes with the highest GTEE. SVDquartets (dashed red) and unpartitioned concatenation analysis CA-ML using RAxML (dashed purple), are shown without filtering. Lines indicate the mean across all replicates, and filled regions indicate the standard error. Rows show three levels of ILS, and columns show two levels of GTEE. When ILS was sufficiently low, gene filtering (up to 75% of genes) increased the accuracy of gene tree summary methods (a-b). When ILS was high to very high, gene filtering had little impact on the accuracy of summary methods or else reduced summary method accuracy (d-f).

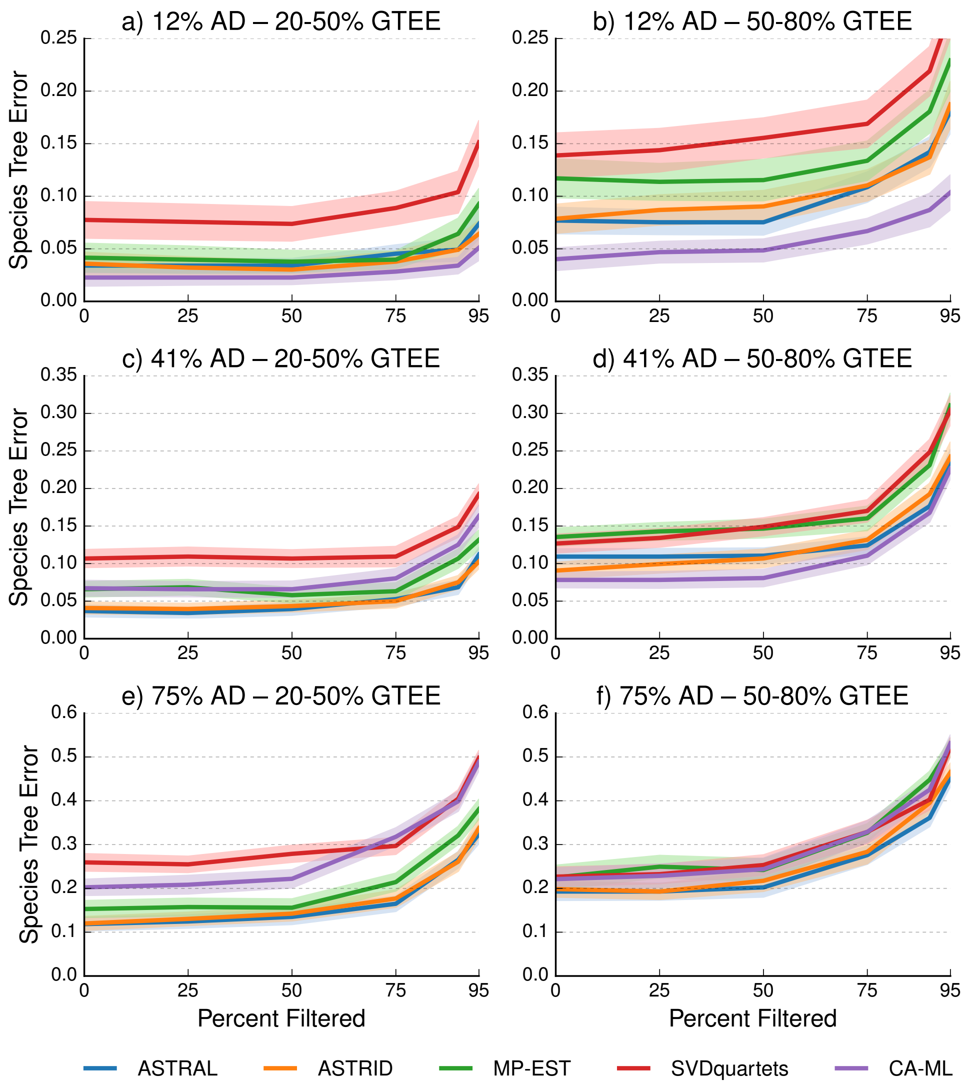


Figure 3.4: **Impact of filtering based on missing data.** The impact of filtering genes by missing data on species tree error (RF error rate) is shown for five methods: ASTRAL-II (blue), ASTRID (orange), MP-EST (green), SVDquartets (red), and CA-ML using RAxML (purple). Genes were filtered by removing the top 25%, 50%, 75%, 90%, and 95% of genes with the highest fractions of missing data; this resulted in removing genes missing at least 50%, 25%, 10%, 5%, and 1% of species. Lines indicate the mean across all replicates, and filled regions indicate the standard error. Rows show three levels of increasing ILS, and columns show two levels of GTEE. Gene filtering based on missing data was at best neutral but often reduced the accuracy of methods.

3.7 TABLES

This section contains two tables: one presented in Section 3.1 Introduction and one presented in Section 3.3 Results.

Table 3.1: **Empirical statistics for biological datasets.** Means and standard deviations across all genes are reported for the following quantities: MSA length, number of *parsimony-informative* sites (sites with at least two character states appearing at least twice), mean bootstrap support, and normalized RF distance between a gene tree and the species tree estimated via CA-ML. Empirical statistics were computed using the MSAs, bootstrap gene trees, and CA-ML trees provided by these studies. In particular, we used the bootstrap gene trees (for each gene) to build a greedy consensus tree and a *SMC* tree (all bipartitions in the SMC tree are induced by greater than 50% of the bootstrapped trees). The greedy consensus tree was used to compute mean bootstrap support by averaging the bootstrap support values across all branches. The SMC tree was used to compute the distance between a gene tree and the species tree computed using CA-ML. The distance between the SMC tree and the estimated species tree is separated into False Positive (FP) rate and False Negative (FN) rate. FN rate is the fraction of branches in the estimated species tree that are missing from the SMC tree; FP rate is the fraction of branches in the SMC tree that are missing from the CA-ML tree. Note that Streicher *et al.* [171] provided concatenated alignments but not alignments for the individual genes in their Dryad repository, so the mean gene length and the mean number of parsimony informative sites per gene could not be computed.

	Dataset	Number	Number	Gene	Number of	Mean bootstrap	SMC versus CA-ML	
Study	type	of species	of gene	length	informative sites	support	FP	FN
Blom *et al.* [54]	exon	29	1361	384 ± 251	12 ± 10	0.28 ± 0.11	0.45 ± 0.26	0.87 ± 0.09
Hosner *et al.* [51]	UCE	91	4817	462 ± 369	37 ± 42	0.28 ± 0.12	0.37 ± 0.25	0.83 ± 0.17
Jarvis *et al.* [23]	exon	48	8251	1612 ± 1308	453 ± 418	0.26 ± 0.07	0.22 ± 0.25	0.85 ± 0.08
Jarvis *et al.* [23]	intron	48	2516	7654 ± 8539	4315 ± 4654	0.47 ± 0.12	0.20 ± 0.13	0.68 ± 0.09
Jarvis *et al.* [23]	UCE	48	3679	2509 ± 164	1062 ± 278	0.40 ± 0.05	0.11 ± 0.11	0.71 ± 0.03
Streicher *et al.* [171]	UCE	29	4784	NA	NA	0.39 ± 0.09	0.36 ± 0.26	0.78 ± 0.12

Table 3.2: **Proportions of replicates for which filtering based on GTEE increased or decreased the accuracy of ASTRAL-II.** The proportion of replicates for which filtering based on GTEE increased and decreased the accuracy of ASTRAL-II is given on the left and right of the forward slash, respectively. The larger of these two values is in bold. If these two fractions do not sum to one, then the remainder is the proportion of replicates for which filtering did not impact accuracy. The number of replicates as well as the mean (± standard deviation) of parsimony informative sites is given for each model condition, specified by the level of ILS and the range of mean GTEE.

Mean GTEE	Number of replicates	Number of informative sites	Proportion of replicates affected by filtering (increased / decreased accuracy) when the following percentages of genes were removed				
			25%	50%	75%	90%	95%
Low/Moderate ILS (12% AD)							
0-20%	10	596 ± 224	0.00 / 0.00	0.00 / 0.00	0.00 / **0.10**	0.00 / 0.00	0.00 / **0.20**
20-50%	23	464 ± 276	**0.09** / 0.04	**0.26** / 0.00	**0.35** / 0.09	**0.30** / 0.04	**0.26** / 0.13
50-80%	26	63 ± 14	**0.23** / 0.04	**0.23** / 0.04	**0.31** / 0.12	**0.35** / 0.23	**0.35** / 0.31
80-85%	5	40 ± 28	0.40 / **0.60**	0.40 / 0.40	0.40 / **0.60**	0.20 / **0.60**	0.20 / **0.80**
85-100%	2	5 ± 3	**0.50** / 0.00	**1.00** / 0.00	**1.00** / 0.00	**1.00** / 0.00	**1.00** / 0.00
High ILS (41% AD)							
0-20%	2	487 ± 9	0.00 / **0.50**	0.00 / **0.50**	0.00 / **1.00**	0.00 / 0.00	0.00 / **0.50**
20-50%	33	349 ± 208	0.03 / **0.09**	0.15 / **0.27**	0.15 / **0.33**	0.03 / **0.55**	0.03 / **0.70**
50-80%	35	42 ± 17	0.11 / **0.17**	**0.26** / 0.17	0.26 / **0.34**	0.14 / **0.49**	0.17 / **0.60**
80-85%	4	22 ± 10	**0.75** / 0.00	**0.50** / 0.00	**0.25** / 0.00	0.25 / 0.25	0.25 / **0.50**
85-100%	1	6 ± 0	0.00 / 0.00	**1.00** / 0.00	**1.00** / 0.00	**1.00** / 0.00	**1.00** / 0.00
Very high ILS (75% AD)							
0-20%	0	NA	NA	NA	NA	NA	NA
20-50%	29	213 ± 123	0.07 / **0.45**	0.07 / **0.59**	0.14 / **0.59**	0.00 / **0.93**	0.00 / **1.00**
50-80%	23	30 ± 9	0.22 / **0.35**	0.17 / **0.61**	0.09 / **0.70**	0.04 / **0.87**	0.00 / **1.00**
80-85%	8	17 ± 10	**0.75** / 0.00	**0.62** / 0.25	**0.62** / 0.25	**0.50** / 0.38	0.00 / **0.75**
85-100%	9	6 ± 6	**0.56** / 0.00	**0.67** / 0.00	**0.78** / 0.00	**0.78** / 0.11	**0.78** / 0.22

CHAPTER 4: DIVIDE-AND-CONQUER PIPELINES WITH NJMERGE

This chapter contains material previously published in "Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge" [220], which was joint work with T. Warnow. NJMerge *is freely available on Github: github.com/ekmolloy/njmerge. Datasets and software commands necessary to reproduce this study are freely available on the Illinois Data Bank: doi.org/10.13012/B2IDB-1424746_V1 and doi.org/10.13012/B2IDB-0569467_V2. All supplementary tables and figures referenced in this chapter are freely available at* Algorithms for Molecular Biology *online: doi.org/10.1186/s13015-019-0151-x. Note that plots and tables appear at the end of this chapter in Sections 4.7 and 4.8, respectively.*

4.1 INTRODUCTION

In Chapter 3, we presented the results of benchmarking species tree estimation methods, including ASTRAL, SVDquartets, and CA-ML using RAxML, on datasets with 26 species and 1 000 genes. All three of these methods can be computationally intensive on large datasets with thousands of species and thousands of genes (Section 2.7), and in this chapter, we address how to scale these methods to larger datasets using divide-and-conquer.

Divide-and-conquer approaches have been developed in the context of phylogeny estimation, for example the family of disk covering methods [221, 222, 223, 224]. Such methods operate by (i) dividing the species set into overlapping subsets, (ii) estimating trees on the subsets, and then (iii) using a supertree method to combine the subset trees into a tree on the entire species set. While supertree methods can provide good accuracy (i.e., retain much of the accuracy in the subset trees) under some conditions, many of the most accurate supertree methods are heuristics for NP-hard optimization problems (Section 2.3); these methods can be computationally intensive on large datasets. Note there are other major challenges to building supertrees, for example when there are large terraces of equally optimal solutions [225]; see also [80].

A supertree meta-method, SuperFine [226], was developed to address the issue of scalability and was later incorporated into step (iii) of a divide-and-conquer pipeline, DACTAL [227]. To the best of our knowledge, only one study [196] has explored using DACTAL for species tree estimation. Specifically, Bayzid *et al.* [196] used DACTAL to speed-up MP-EST on multi-locus datasets with at most 37 species and showed that this approach is statistically consistent under the Multi-Species Coalescent (MSC) model [196]. Their code was not publicly available at the time of our study (Md. S. Bayzid, personal communication with T. Warnow). As recently reviewed by Warnow [56], divide-and-conquer continues to be challenging, and yet Bininda-Emonds and many others [55, 228] have argued that divide-and-conquer is a promising and necessary approach for estimating the Tree of Life.

We propose a new divide-and-conquer approach for scaling phylogeny estimation methods to larger datasets that operates by (i) dividing the species set into pairwise disjoint subsets, (ii) estimating a tree on each of subset, and then (iii) *merging* the subset trees into a phylogeny on the full species set. Note that the term merge refers to building a (refined) compatibility supertree (Definition 2.7). While a (refined) compatibility supertree is guaranteed to exist when the subset trees are on pairwise disjoint species sets, the subset tree topologies contain no information about how to perform this merger (Figure 4.1). Indeed, *disjoint tree mergers (DTMs)* require a set $\mathcal{A}$ of auxiliary data, so DTM methods can be viewed as constrained phylogeny estimation: estimate a phylogeny T from data given in $\mathcal{A}$ subject to the topological constraints implied by the input set $\mathcal{T}$ of subset trees. This does not require the trees in $\mathcal{T}$ to be *edge separable* for T, meaning that every tree in $\mathcal{T}$ can be obtained from T through a sequence of zero or more edge deletions or conversely that T can be obtained by connecting the trees in $\mathcal{T}$ by edges (Figure 4.1). For example, the *caterpillar tree* on $\{A, B, C, D, \ldots, H\}$ obtained by making a path with the leaves hanging off it in alphabetical order is a compatibility supertree for $\mathcal{T} = \{AC|EG,\ BD|FH\}$, and yet the trees in $\mathcal{T}$ are not edge separable for the caterpillar tree.

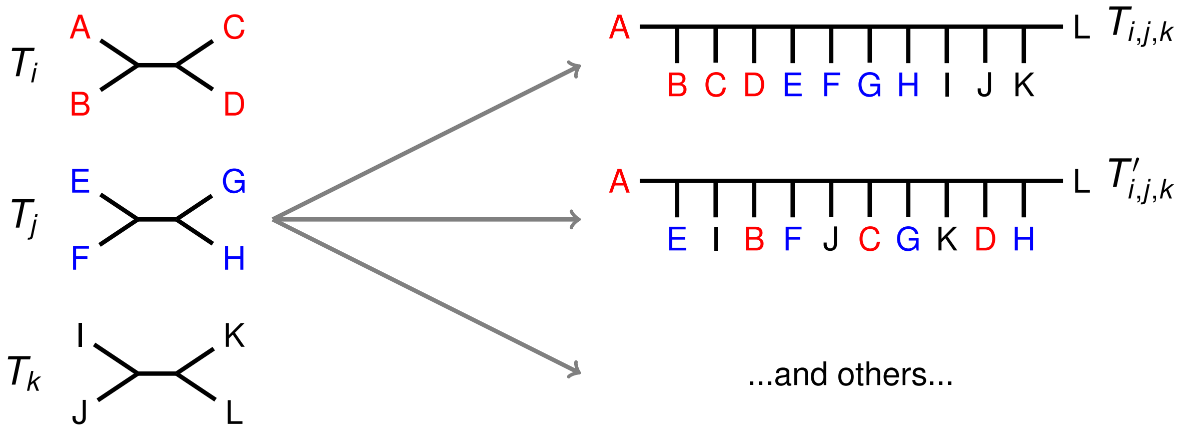


Figure 4.1: **Compatibility Supertree Example.** Two compatibility supertree for $\mathcal{T} = \{T_i, T_j, T_k\}$ are shown. Notably, the trees in $\mathcal{T}$ are edge separable for $T_{i,j,k}$ but are not edge separable for $T'_{i,j,k}$.

In the remainder of this chapter, we present the first DTM method, *NJMerge*. As we will show, NJMerge runs in polynomial time and enables divide-and-conquer pipelines (for estimating gene trees or species trees) that are provably statistically consistent. We also present the results of a simulation study evaluating the effectiveness of NJMerge for multi-locus species tree estimation. Our study found that NJMerge improves the running time of ASTRAL-III, SVDquartets, and CA-ML using RAxML without sacrificing accuracy. Furthermore, NJMerge enabled SVDquartets and RAxML to run on large datasets (e.g., 1 000 species and 1 000 genes), on which SVDquartets and RAxML would otherwise fail to run when limited to 64 GB of memory. While NJMerge is not guaranteed to find a compatibility supertree, the failure rate of NJMerge in our experiments was low (less than 1% of tests), so NJMerge failed on fewer datasets than either ASTRAL-III, SVDquartets, or RAxML when given the same computational resources. Together, these empirical and theoretical results suggest that NJMerge is a valuable technique for species tree estimation, especially when computational resources are limited.

4.2 APPROACH

In this section, we present the NJMerge algorithm and then show how it can be used within a divide-and-conquer framework to enable statistically consistent phylogeny estimation pipelines.

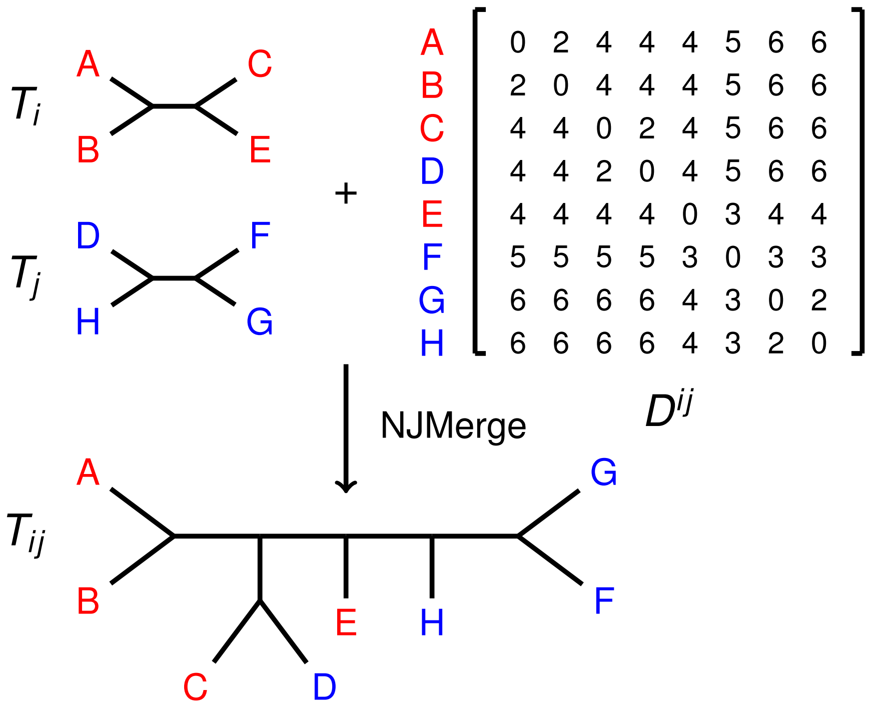
4.2.1 NJMerge

NJMerge extends NJ by imposing a set of topological constraints on the output tree; therefore, NJMerge has the same input as NJ but additionally requires a set of constraint trees.

- **NJMerge Input**:
  - Set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of unrooted phylogenetic trees such that $S(T_i) \cap S(T_j) = \emptyset$ for all $i \neq j$
  - Set $\mathcal{A}$ of auxiliary data; specifically
    - An $n \times n$ dissimilarity matrix D on label set $S = \cup_{i=1}^{k} S(T_i)$
- **NJMerge Output**: A (possibly refined) compatibility supertree for $\mathcal{T}$ that is fully resolved

Because the trees in $\mathcal{T}$ are on pairwise disjoint leaf label sets, we refer to them as being *leaf-label-disjoint*. A compatibility supertree always exists for leaf-label-disjoint trees; however, the objective is to find a tree that is close to the true (but unknown) phylogeny from the set of all compatibility supertrees for $\mathcal{T}$. NJMerge uses the dissimilarity matrix D to achieve this objective (Figure 4.2).

The traditional NJ algorithm has an iterative design that builds the tree from the bottom up, producing a rooted tree that is then unrooted; this approach is akin to hierarchical clustering. Initially, there are n leaf labels. When two leaf labels x and y are selected to be siblings, the pair of leaf leaves is replaced by a single leaf label z, which represents a rooted tree on the two leaves. This reduces the number of leaf labels by one. This process repeats until there is only one leaf, representing a rooted tree on n leaves; this tree is then unrooted and returned. Note that at each iteration, NJ makes a siblinghood decision by building a secondary matrix Q from D and simultaneously searching for a minimal element. If $Q[i, j]$ is a minimal element of Q, leaves i and j are made siblings; see [121] for details.



$$\begin{array}{c|cccccccc}
A & 0 & 2 & 4 & 4 & 4 & 5 & 6 & 6 \\
B & 2 & 0 & 4 & 4 & 4 & 5 & 6 & 6 \\
C & 4 & 4 & 0 & 2 & 4 & 5 & 6 & 6 \\
D & 4 & 4 & 2 & 0 & 4 & 5 & 6 & 6 \\
E & 4 & 4 & 4 & 4 & 0 & 3 & 4 & 4 \\
F & 5 & 5 & 5 & 5 & 3 & 0 & 3 & 3 \\
G & 6 & 6 & 6 & 6 & 4 & 3 & 0 & 2 \\
H & 6 & 6 & 6 & 6 & 4 & 3 & 2 & 0
\end{array}$$

Figure 4.2: **NJMerge Input/Output Example.** The input to NJMerge is a set $\mathcal{T} = \{T_i, T_j\}$ of constraint trees and a dissimilarity matrix D^{ij} that is additive (Definition 2.11) for the tree T given by the Newick string: $(((A, B), (C, D)), E, (F, (G, H)))$ (note that T has the same topology as $T_{i,j}$ but swaps leaves H and F). Traditional NJ applied to D^{ij} returns T (Theorem 2.2); however, NJMerge rejects the siblinghood proposal (G, H), because it violates constraint tree T_j. NJMerge makes G and F siblings, returning tree $T_{i,j}$.

NJMerge operates in a similar fashion even using the same formulas as NJ to compute Q

and update D; however, NJMerge can make different siblinghood decisions than NJ based on the input constraint trees. After accepting a siblinghood proposal, NJMerge updates D as well as the constraint trees in $\mathcal{T}$. For example, if x labels a leaf in T_i and y labels a leaf in T_j, then the siblinghood decision (x, y) requires T_i and T_j to be updated by relabeling x in T_i and y in T_j by a new label z, which represents the rooted subtree (x, y). Because siblinghood decisions can result in constraint trees no longer being on pairwise disjoint leaf label sets (Figure 4.3), they have the potential to make the set of constraint trees incompatible. Determining whether a set of $k > 2$ unrooted trees is compatible (Definition 2.7) is NP-complete [71, 229], so NJMerge uses a polynomial-time heuristic.

At each iteration, NJMerge sorts the entries of Q from least to greatest and then, based on this ordering, evaluates each siblinghood proposal (x, y) as follows.

- *Test that the proposed siblinghood does not violate the constraint trees.*
  - If x and y both label leaves in some constraint tree T_i, check that they label siblings in T_i. If not, move to the next siblinghood proposal.
- *Use a heuristic to test that the proposed siblinghood does not make the set of constraint trees incompatible.*
  - Update all of the constraint trees as follows. If x and y both label leaves in a constraint tree T_i, replace (x, y) by a single leaf labeled z. If only x (or y) are in some constraint T_i, then update T_i relabeling x (or y) with the new label z.
  - Use a heuristic to test if $\mathcal{T}$ is compatible. If the test passes, accept the proposal; otherwise, reverse the updates and move to the next proposal.

Because a heuristic is used to test compatibility, it is possible for NJMerge to accept a siblinghood proposal that will eventually cause the algorithm to fail when none of the remaining leaves can be joined without violating the compatibility of constraint trees. Although NJMerge can fail, it is easy to see that when NJMerge returns a tree, it is a compatibility supertree for the input set $\mathcal{T}$ of constraint trees.

If a set $\mathcal{T}$ of trees is compatible, then every pair of trees in $\mathcal{T}$ is compatible (the reverse statement does not hold). We implemented NJMerge using pairwise compatibility as a heuristic. It is worth noting that NJMerge was developed as a subroutine of TreeMerge (Chapter 5), and in this context, NJMerge is run on $k = 2$ constraint trees of bounded size. In retrospect, it suffices to check only those pairs of constraint trees with leaves labeled by at least one of x and y; all other pairs of trees are unchanged by accepting the siblinghood proposal and are pairwise compatible by induction. After being updated, this subset of

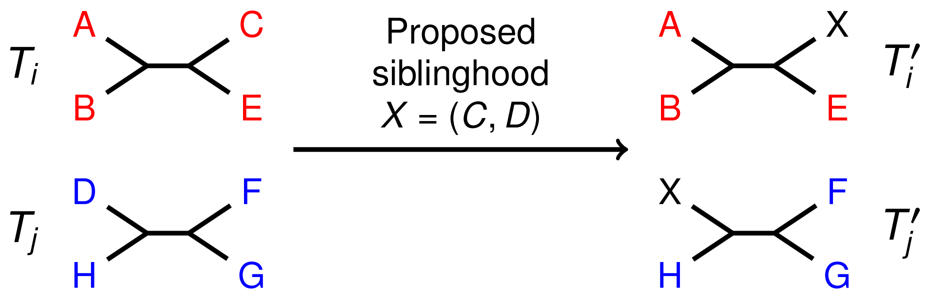


Figure 4.3: **NJMerge Siblinghood Proposal Example.** In this example, NJMerge evaluates the siblinghood proposal (C, D). Because C labels a leaf in T_i and D labels a leaf in T_j, NJMerge updates the constraint trees T_i and T_j based on the proposed siblinghood, labeling C and D by X, which represents the siblinghood (C, D). The two updated constraint trees are no longer on disjoint label sets.

constraint trees can be rooted at the edge incident to the leaf labeled z, which represents subtree (x, y). Testing the compatibility of rooted trees can be accomplished in polynomial time [230, 231]. Therefore, an alternative heuristic is to test the compatibility of all constraint trees with the new leaf label.

Theorem 4.1. Let $\mathcal{T} = \{T_i\}_{i=1}^k$ be a set of unrooted phylogenetic tree, and let D be an $n \times n$ dissimilarity matrix on label set $S = \bigcup_{i=1}^k S(T_i)$. Then, NJMerge applied to input $(\mathcal{T}, D)$ fails or returns a (possibly refined) compatibility supertree for $\mathcal{T}$ in $O(n^4 k)$ time if using the pairwise compatibility heuristic or in $O(n^4 \log^2 n)$ time if using the alternative heuristic.

Proof. We first prove that if NJMerge completes, then it returns a (possibly refined) compatibility supertree. At each iteration, NJMerge updates every constraint tree $T_i \in \mathcal{T}$ based on the siblinghood descision. We use superscript (w) to denote a tree after the w^{th} siblinghood decision, where each leaf representing a rooted subtree is replaced by that subtree. Initially, $T_i^{(0)}$ is compatible with T_i, as no siblinghood decisions have been made (Definition 2.1). Assume that after $w-1$ siblinghood decisions, $T_i^{(w-1)}$ is compatible T_i. At the w^{th} siblinghood decision, there are four cases:

- Neither x nor y label leaves in $T_i^{(w-1)}$. NJMerge does nothing. In this case, $T_i^{(w)} = T_i^{(w-1)}$, so $T_i^{(w)}$ is compatible T_i.
- Leaves labeled by x and y are siblings in $T_i^{(w-1)}$. NJMerge replaces (x, y) by a single leaf labeled z. In this case, $T_i^{(w)} = T_i^{(w-1)}$, so $T_i^{(w)}$ is compatible with T_i.
- Only x labels a leaf in $T_i^{(w-1)}$. NJMerge relabels x by z. In this case, $T_i^{(w)}$ is $T_i^{(w-1)}$ with

the subtree represented by y connected to the branch above the subtree represented by x, so $T_i^{(w)}$ is compatible with $T_i^{(w-1)}$ and thus is compatible with T_i.

- Only y labels a leaf in $T_i^{(w-1)}$. Apply the same argument from above.

By induction on the number of siblinghood decisions, $T_i^{(n-1)}$ is compatible with T_i for all $T_i \in \mathcal{T}$. After $n-1$ iterations, every $T_i \in \mathcal{T}$ is replaced by a single leaf, which represents a rooted tree on n leaves, so $T_1^{(n-1)} = T_2^{(n-1)} = \cdots = T_k^{(n-1)} = T$. Therefore, the unrooted version of T is a refined compatibility supertree for $\mathcal{T}$.

We now address the worst-case running time of NJMerge (Algorithm 4.2 in Section 4.9), assuming that n species are divided into k subsets of size n/k for simplicity. To begin, we compute a vector $\vec{r}$ containing the row sums of D in $O(n^2)$ time. Then, we perform $O(n)$ iterations with each iteration having three steps.

- In step 1, we build and sort the $O(n^2)$ entries of Q from least to greatest. Each element of Q can be computed in constant time from D and $\vec{r}$, so this takes $O(n^2 \log n)$ time.

- In step 2, we evaluate each siblinghood proposal $z = (x, y)$ in the order suggested by sorting Q, breaking on the first siblinghood proposal that passes the heuristic test of compatibility, denoted `IsCompatibleHeuristic` in Algorithm 4.3 (Section 4.9).

  - *Pairwise compatibility heuristic:* Test the compatibility of all pairs of constraint trees with the new leaf label z are compatible (Algorithm 4.4 in Section 4.9). This can be achieved by restricting the two trees to their shared leaf label set and then computing their Robinson-Foulds (RF) distance using the linear-time algorithm proposed by Day [232]. Each of the k trees in $\mathcal{T}$ has at most n/k leaves, so this takes $O(nk)$ time.

  - *Alternative heuristic:* Test the compatibility of all constraint trees with the new leaf label z using the algorithm by Deng and Fernández-Baca [231], which runs in $O(M \log^2 M)$ time, where M is the total number of edges and nodes in $\mathcal{T}$. Each of the k trees in $\mathcal{T}$ has at most n/k leaves, so this takes $O(n \log^2 n)$ time.

  In the best case, the first element passes the heuristic test, and in the worst case, none of the $O(n^2)$ proposals pass the heuristic test (in which case NJMerge fails). Therefore, in the worst-case, step 2 requires $O(n^3 k)$ time if using the pairwise compatibility heuristic and $O(n^3 \log^2 n)$ time if using the other heuristic.

- In step 3, we update D and $\vec{r}$ in $O(n)$ time.

In summary, the worst-case running time of NJMerge is $O(n^4 k)$ if using the pairwise compatibility heuristic and is $O(n^4 \log^2 n)$ if using the alternative heuristic. QED.

Note that if the best case always occurs for step 2, then the running time is $O(n^3 \log n)$; therefore, the running time of NJMerge can vary greatly depending on the input. Although running time improves in this scenario, it means that NJMerge returns the same tree as the traditional NJ algorithm, which is less than ideal, as this means that NJMerge does not improve upon the accuracy of traditional NJ.

4.2.2 Divide-and-Conquer Pipelines for Phylogeny Estimation

NJMerge can be used in divide-and-conquer pipelines for phylogeny estimation, as shown in Algorithm 4.1 and Figure 4.4.

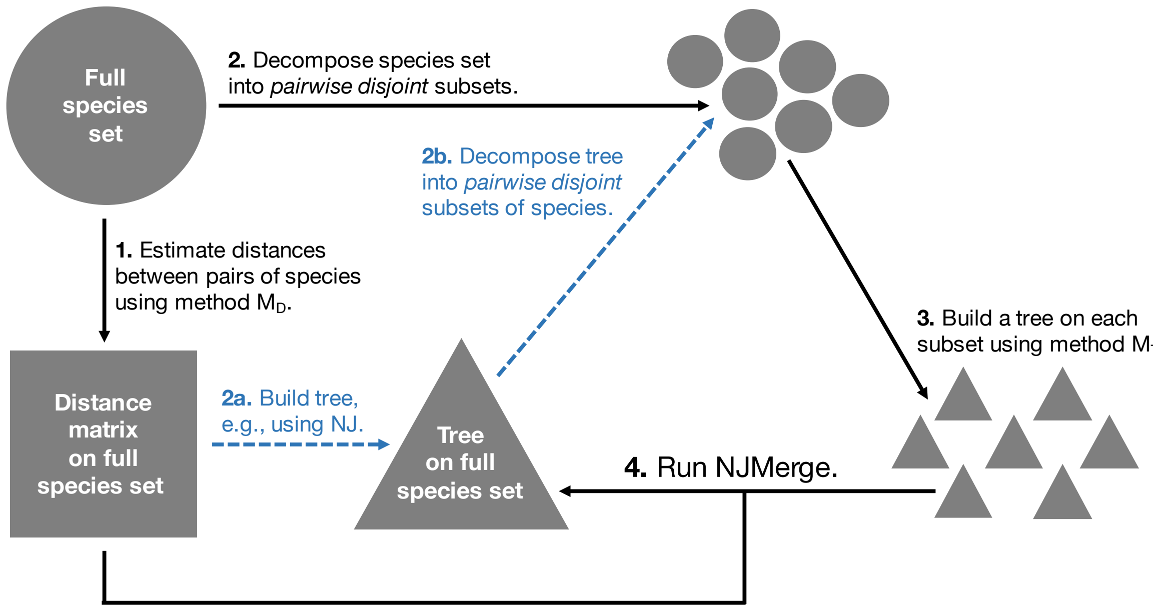

Full species set
2. Decompose species set into *pairwise disjoint* subsets.
2b. Decompose tree into *pairwise disjoint* subsets of species.
1. Estimate distances between pairs of species using method M_D.
3. Build a tree on each subset using method M_T.
Distance matrix on full species set
2a. Build tree, e.g., using NJ.
Tree on full species set
4. Run NJMerge.

Figure 4.4: **Divide-and-Conquer Pipeline using NJMerge.** Circles are sets of (species) labels, squares are dissimilarity matrices, and triangles are phylogenetic trees. Note that henceforth M_D (in step 1) is referred to as Φ_D and M_T (in step 3) is referred to as Φ_T.

4.2.3 Statistical Consistency

Algorithm 4.1 shows that a user must select a method for estimating a dissimilarity matrix Φ_D (step 1), a method Φ_0 for dividing the leaf labels of a tree into pairwise disjoint

Algorithm 4.1: Divide-and-Conquer Pipeline using NJMerge.

```
Input :
     $S$  Set of species labels
     $X$  Dataset for $S$
  $\Phi_D$  Method for estimating distances between pairs of species in $X$
  $\Phi_T$  Method for estimating phylogeny from $X$
  $\Phi_0$  Method for dividing a label set into pairwise disjoint subsets of bounded size given a tree on that
        label set (e.g., the centroid edge decomposition [233])
   $max$  Maximum subset size
Output: A (possibly refined) compatibility supertree for $\mathcal{T}$

Function DivideAndConquer($X$, $\Phi_T$, $\Phi_D$, $\Phi_0$, $max$):
    Step 1: Estimate distances between all pairs of species.
    $D \leftarrow \Phi_D(X)$

    Step 2: Divide the species set into pairwise disjoint subsets.
    $T_0 \leftarrow$ NJ($D$)
    $\{S_1, S_2, \ldots, S_k\} \leftarrow \Phi_0(T_0, max)$

    Step 3: Estimate a tree on the dataset restricted to each subset of species, producing a set of
    constraint trees; this can performed in serial or in parallel, depending on the computational
    resources available.
    $\mathcal{T} \leftarrow \emptyset$
    for $i \in 1, 2, \ldots, k$ do
        $T_i \leftarrow \Phi_T(X|_{S_i})$
        $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_i\}$
    end

    Step 4: Merge subset trees.
    $T \leftarrow$ NJMerge($\mathcal{T}$, $D$)

    return $T$
```

subsets (step 2), a maximum subset size *max* (step 2), and a method Φ_T for estimating subset trees (step 3). Therefore, users can select methods Φ_D and Φ_T to be appropriate for gene tree estimation or species tree estimation (note that the user also should select *max* to be appropriate given these methods and the computational resources that they can access). We prove that with the proper choices of Φ_D and Φ_T divide-and-conquer pipelines using NJMerge are statistically consistent under the Generalized Time Reversible (GTR) model of DNA evolution and under the MSC model. These results follow from Theorem 4.2.

Theorem 4.2. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of unrooted phylogenetic trees, and let D be an $n \times n$ dissimilarity matrix on label set $S = \bigcup_{i=1}^{k} S(T_i)$, and let T^* be an unrooted, fully resolved tree on label set S. Suppose that D is a nearly additive matrix for T^* (Definition 2.11) and that T^* is compatible with T_i for all $i \in \{1, \ldots, k\}$ (Definition 2.1). Then, NJMerge applied to input $(\mathcal{T}, D)$ returns T^*.

Proof. NJ applied to a nearly additive dissimilarity matrix for T^* will return T^* (Theorem 2.2). Because T^* is compatible with every tree in $\mathcal{T}$, the siblinghood proposals suggested by NJ will never violate the trees in $\mathcal{T}$ or the (heuristic test of the) compatibility of $\mathcal{T}$. It follows that NJMerge applied to $(\mathcal{T}, D)$ will return the same output as NJ applied to D, which is T^*. QED.

We now define statistical consistency in the context of gene tree estimation (Definition 4.1) and show that NJMerge can be used to create statistically consistent divide-and-conquer pipelines for gene tree estimation (Corollary 4.1).

Definition 4.1 (Statistical Consistency under GTR model)**.** Let (T, Θ) be a GTR model tree with topology T and numerical parameters Θ. A method M for constructing gene trees from DNA sequences is statistically consistent under the GTR model if, for all $\epsilon > 0$, there exists a constant $l > 0$ such that, when given at least l sites generated independently from the GTR model tree, M returns the unrooted version of T with probability at least $1 - \epsilon$.

Corollary 4.1. NJMerge can be used in a gene tree estimation pipeline that is statistically consistent under the GTR model.

Proof. Let (T^*, Θ) be a GTR model tree, let Φ_D be a method for calculating distances between pairs of DNA sequences, and let Φ_T be a method for constructing trees from site patterns (DNA sequences). Suppose that

- the divide-and-conquer pipeline produces k pairwise disjoint subsets of DNA sequences labeled by the set S
- method Φ_D is statistically consistent under the GTR model, for example the log-det distance [123])
- method Φ_T is statistically consistent under the GTR model, for example a maximum likelihood (ML) method [116]

Now let $\epsilon > 0$, and select $\epsilon_D, \epsilon_T > 0$ such that $\epsilon_D + k\epsilon_T < \epsilon$. By Definition 4.1, there exists a constant l_D such that NJ applied to matrix D computed from DNA sequences with at least l_D sites returns T^* with probability at least $1 - \epsilon_D$, and there exists a constant l_T such that Φ_T given DNA sequences with at least l_T sites returns T^* with probability at least $1 - \epsilon_T$. If a phylogenetic distance matrix D is calculated using Φ_D and a set of k phylogenetic (constraint) trees $\mathcal{T}$ are constructed using Φ_T, given DNA sequences with at least $\max\{l_D, l_T\}$ sites, then the probability that NJ applied to D returns T^* and that Φ_T returns a tree that agrees with T^* for all k constraint trees in $\mathcal{T}$ is at least $1 - \epsilon$, as

$$\begin{aligned}(1-\epsilon_D)(1-\epsilon_T)^k &\geq (1-\epsilon_D)(1-k\epsilon_T) \quad \text{by Bernoulli's Inequality [234]} \\ &= 1 - \epsilon_D - k\epsilon_T + k\epsilon_D\epsilon_T \\ &> 1 - (\epsilon_D + k\epsilon_T) > 1 - \epsilon\end{aligned} \tag{4.1}$$

Then, by Theorem 4.2, NJMerge applied to the input $(\mathcal{T}, D)$ will return the T^* with probability at least $1 - \epsilon$, and by Definition 4.1, NJMerge is statistically consistent under the GTR model. QED.

Finally, we define statistical consistency in the context of species tree estimation (Definition 4.2) and show that NJMerge can be used to create statistically consistent divide-and-conquer pipelines for species estimation (Corollary 4.2).

Definition 4.2 (Statistical Consistency under MSC model)**.** Let (T, Θ) be an MSC model tree with topology T and numerical parameters Θ. A method M for constructing species trees from true gene trees is statistically consistent under the MSC model if, for all $\epsilon > 0$, there exists a constant $m > 0$ such that, given at least m gene trees generated independently from the MSC model tree, M returns T with probability at least $1 - \epsilon$.

Corollary 4.2. NJMerge can be used in a species tree estimation pipeline that is statistically consistent under the MSC model.

Proof. This proof is similar to Corollary 4.1 except that we must make choices appropriate for species trees estimation. Therefore, we would suppose that

- the divide-and-conquer pipeline produces k pairwise disjoint subsets of species labeled by the set S
- method Φ_D is statistically consistent under the MSC model, for example the AGID [153]
- method Φ_T is statistically consistent under the MSC model, for example ASTRAL [49]

and then, letting $\epsilon > 0$, we would select $\epsilon_D, \epsilon_T > 0$ such that $\epsilon_D + k\epsilon_T < \epsilon$ and proceed in a fashion similar to Corollary 4.1. QED.

Note that the above proof easily could be extended to the MSC+GTR model for site-based methods that take the concatenated alignment as input.

Lastly, because distance estimation (for D) and phylogeny estimation (for $\mathcal{T}$) are both performed using statistically consistent methods, this begs the question: why not just run NJ instead of NJMerge? The major take-away is that data are neither infinite nor error-free in practice, so two statistically consistent methods can perform very differently (in terms of accuracy) on the same dataset.

4.3 PERFORMANCE STUDY

We present results of using NJMerge to estimate species trees on large multi-locus datasets simulated using the protocol presented in [152]. Our simulation produced four model conditions, described by two numbers of species (100 and 1 000) and two levels of ILS (low/moderate and very high), each with 20 replicate datasets. Datasets included both exon-like sequences and intron-like sequences with exons characterized by slower rates of evolution across sites (less phylogenetic signal) and introns characterized by faster rates of evolution across sites (greater phylogenetic signal). The 100-species datasets were analyzed using 25, 100, and 1 000 genes, and the 1 000-species datasets were analyzed using 1 000 genes; note that exons and introns were always analyzed separately. For each of these 320 datasets, we estimated dissimilarity matrices using two different methods and constraint trees using four different methods. This provided 2 560 different tests on which to evaluate NJMerge. NJMerge failed on 11/2 560 tests, so the failure rate in our experiments was less than 1% (although these tests are not independent). Species tree estimation methods were evaluated in terms of running time and species tree error, as measured by the RF error rate (Equation 2.3).

4.3.1 Simulated Datasets

Species trees and gene trees: Datasets, each with a true species tree and 2000 true gene trees, were simulated under the MSC model using SimPhy version 1.0.2. All model conditions had deep speciation (towards the root) and 20 replicate datasets. By holding the effective population size (EPS) constant (200K) and varying the species tree height (in generations), model conditions with different levels of incomplete lineage sorting (ILS) were generated. For species tree heights of 10M and 500K generations, the average distance (AD) was 8–10% and 68–69% respectively. Thus, we referred to these levels of ILS as "low/moderate" and "very high," respectively.

DNA (gene) sequence data: DNA sequences were simulated for each true gene tree using INDELible version 1.03 [197] under the GTR+GAMMA model. For each gene, the parameters for the GTR+GAMMA model ($\vec{\pi}$, Q, and α) were drawn from distributions based on estimates of these parameters from the Avian Phylogenomics Dataset [23]. Distributions were fitted for exons and introns, separately (Supplementary Table S1), so for each dataset (with 2000 genes), 1 000 gene sequences were simulated with parameters drawn from the exon distributions, and 1 000 gene sequences were simulated with parameters drawn from the intron distributions. Sequence lengths were also drawn from a distribution (varying from 300 to 1 500 bp). DNA sequences were simulated without insertions or deletions, so multiple sequence alignment (MSA) estimation was not necessary.

Estimated gene trees: ML gene trees were estimated using FastTree-2 version 2.1.10 under the GTR+CAT model. GTEE was computed as the normalized symmetric difference between true and estimated gene trees (Equation 2.4), averaged across all gene trees. Across all model conditions datasets, the gene tree estimation error (GTEE), averaged across all replicates, ranged from 26% to 51% for introns and 38% to 64% for exons and thus was higher for exon datasets (Supplementary Table S2).

4.3.2 Species Tree Estimation

For each model condition (characterized by number of species and level of ILS), species tree estimation methods were run either given the exon-like genes or the intron-like genes as input. Species trees were estimated on 25, 100, or 1 000 genes for the 100-species species and 1 000 genes for the 1 000-species datasets using three different methods: ASTRAL-III (as implemented in version 5.6.1), SVDquartets (as implemented in PAUP* version 4a161), and CA-ML under the GTR+GAMMA model (as implemented in RAxML version 8.2.12 with pthreads and SSE3).

4.3.3 Divide-and-conquer pipeline using NJMerge

Distance matrices: Distance matrices were created using two different approaches.

- D_{AGID} denotes the AGID matrix computed from estimated gene trees using ASTRID version 1.1.
- D_{LD} denotes the log-det distance matrix computed from concatenated alignment using PAUP* version 4a163.

Subset decomposition: We decomposed the species set into subsets as indicated by the blue dashed arrows in Figure 4.4. First, a starting tree was built by running NJ (as implemented in FastME version 2.1.5) on the estimated dissimilarity matrix and then used to define pairwise disjoint subsets of species. Essentially, the set of species was divided into subsets by repeatedly deleting centroid edges (edges whose deletions divide the leaf set roughly in half) until the resulting subsets are smaller than the predetermined maximum size; see the *centroid edge decomposition* described in PASTA [233]. Datasets with 100 species were decomposed into four to six subsets with a maximum subset size of 30 species, and datasets with 1 000 species were decomposed into 10–15 subsets with a maximum subset size of 120 species.

Constraint trees: Constraint trees were created using four different approaches.

- $\mathcal{T}_{true}$ denotes constraint trees computed by restricting the true species tree to each subset of species.
- $\mathcal{T}_{AST}$ denotes constraint trees estimated by running ASTRAL-III on each subset, i.e., on the estimated gene trees restricted to each subset of species.
- $\mathcal{T}_{SVD}$ denotes constraint trees estimated by running SVDquartets on each subset, i.e., on the concatenated alignment restricted to each subset of species.
- $\mathcal{T}_{RAX}$ denotes constraint trees estimated by running RAxML on each subset, i.e., on the concatenated alignment restricted to each subset of species.

Notation: We often specify the inputs to NJ and NJMerge using the following notation: NJ(D) and NJMerge($\mathcal{T}$, D). For example, NJMerge($\mathcal{T}_{RAX}$, D_{LD}) refers to NJMerge given the RAxML constraint trees and the log-det distance matrix as input, whereas NJMerge($\mathcal{T}_{RAX}$, D) refers to NJMerge given the RAxML constraint trees and *either* the AGID matrix *or* the log-det distance matrix as input.

4.3.4 Computational Experiments and Running Time Evaluation

All computational experiments were run on the Blue Waters supercomputer, specifically the XE6 dual-socket nodes with 64 GB of physical memory and two AMD Interlagos model 6276 CPU processors (i.e., one per socket each with 8 floating-point cores). All methods were given access to 16 threads with 1 thread per bulldozer (floating-point) core. SVDquartets and RAxML were explicitly run with 16 threads; however, ASTRAL-III and NJMerge were not implemented with multi-threading at the time of this study. All methods were restricted to a maximum wall-clock time of 48 hours.

Running time was measured as the wall-clock time recorded in seconds. For ASTRAL, SVDquartets, and RAxML, timing data was recorded for running the method on the full dataset as well as for running the method on subsets of the dataset to produce constraint trees for NJMerge. RAxML did not complete within the maximum wall-clock time of 48 hours on datasets with 1 000 species, so we used the last checkpoint file to evaluate species tree estimation error and running time (i.e., running time was the time between the info file being written and the last checkpoint file being written).

We computed the total running time of the NJMerge pipeline by combining the timing data for estimating the dissimilarity matrix, estimating the subset trees, and running NJMerge; see Supplementary Tables S9 and S10 for the average time required for each of these steps. If a user only had access to one compute node, subset trees would need to be estimated in serial, so the running time of the NJMerge pipeline would be

$$time\big(\Phi_D(X)\big) + \sum_{T \in \mathcal{T}} time\big(\Phi_T(X|_{S(T)})\big) + time\big(NJMerge(\mathcal{T}, D)\big) \tag{4.2}$$

using the notation defined in Algorithm 4.1. Note that Equation 4.2 does not include the timing data for estimating the starting tree, as this took less than a minute even for datasets with 1 000 species. Finally, if given access to multiple compute nodes (at least six nodes for the 100-species datasets and at least 15 nodes for the 1 000-species datasets), the subset trees could be estimated in an embarrassingly parallel fashion. In this case, the running time of the NJMerge pipeline would be

$$time\big(\Phi_D(X)\big) + \max_{T \in \mathcal{T}}\{time\big(\Phi_T(X|_{S(T)})\big)\} + time\big(NJMerge(\mathcal{T}, D)\big) \tag{4.3}$$

Results computed using Equation 4.3 are provided in [235].

It is worth noting that running ASTRAL-III and computing the AGID matrix require gene trees to be estimated. Using the same experimental set-up (a single Blue Waters compute node with 64 GB of memory and 16 floating-point cores), FastTree-2 took on average 18 ± 2 minutes to estimate 1 000 gene trees for datasets with 100 species and on average 217 ± 20 minutes to estimate 1 000 gene trees for datasets with 1 000 species (Supplementary Tables S4 and S5). The amount of time for gene tree estimation can vary greatly, depending on the method used and the analysis performed. We did not include the time to estimate gene trees in the reported running times.

4.4 RESULTS

Pipelines using NJMerge can be viewed in two different ways: (i) as techniques for potentially improving the accuracy of NJ (hopefully without a large increase in running time) or (ii) as techniques for potentially improving the scalability of the method used to estimate constraint trees (hopefully without sacrificing accuracy). For the former, we would expect NJMerge to improve the accuracy of traditional NJ whenever distance-based species tree estimation is less accurate than other methods (which can used to estimate constraint trees). For the latter, we would expect NJMerge to improve the running time of methods (used to estimate constraint trees) whenever these methods are more computationally intensive than distance-based methods (which is often the case).

We compared the accuracy of the NJMerge pipeline to traditional NJ, and we also compared the accuracy and running time of the NJMerge pipeline to running Φ_T on the full dataset, where Φ_T is the method used to estimate the constraint trees for NJMerge. Results for intron-like datasets are provided below, and results for exon-like datasets are provided in the Supplementary Materials. Unless otherwise noted, results were similar for both sequence types; however, species trees estimated on the exon datasets had slightly higher error rates than those estimated on the intron datasets. This is expected, as the exons had slower rates of evolution (and thus less phylogenetic signal) than the introns.

4.4.1 How do pipelines using NJMerge compare to NJ?

In this section, we report on the effectiveness of using NJMerge as compared to traditional NJ in terms of species tree accuracy.

Impact of estimated dissimilarity matrix: In this section, we compare traditional NJ to the NJMerge pipeline, both given distance matrices estimated from 100-species datasets with varying numbers of genes (Figure 4.5 and Supplementary Figure S1). Because the accuracy of NJMerge also depends on the amount of error in the input constraint trees, we studied the ideal case, giving NJMerge true constraint trees (i.e., constraint trees that agreed with the true species tree). We found that NJMerge($\mathcal{T}_{true}$, D) was more accurate than NJ(D) for all model conditions and that the difference in error was especially large when the number of genes was small and the level of ILS was very high; for example, the difference in mean error was greater than 15% when matrices were estimated from 25 introns but was closer to 5% when matrices were estimated from 1 000 introns. A similar trend was observed for matrices computed using the log-det distance. Interestingly, both NJ(D) and

NJMerge($\mathcal{T}_{true}$, D) were more accurate when given the AGID matrix rather than the log-det distance matrix as input even when ILS was low/moderate. In summary, NJMerge($\mathcal{T}_{true}$, D) was always more accurate than NJ(D), but the improvement in accuracy was greater under challenging model conditions, suggesting that NJMerge($\mathcal{T}_{true}$, D) is more robust to error in the estimated dissimilarity matrix than NJ(D).

Impact of estimated constraint trees: In this section, we compare traditional NJ to the NJMerge pipeline given estimated constraint trees on datasets with 1 000species and 1 000 genes (Figure 4.6 and Supplementary Figure S2). When ILS was low/moderate, NJMerge outperformed NJ regardless of the method used to estimate species trees. For intron-like datasets with low/moderate ILS, the use of constraint trees reduced the median species tree error from 11–14% (NJ) to less than 3–6% (NJMerge); however, when ILS was very high, the performance of NJMerge varied greatly with the species tree estimation method. Specifically, NJMerge($\mathcal{T}_{SVD}$, D) and NJMerge($\mathcal{T}_{RAX}$, D) were less accurate than NJ(D) by 0-4% on average, whereas NJMerge($\mathcal{T}_{AST}$, D) was more accurate than NJ(D) by 0–1% on average (Supplementary Tables S7 and S8). These trends were consistent with the relative performance of methods on the 100-species datasets (Figure 4.7 and Supplementary Figure S3). In summary, NJMerge was highly impacted by the quality of the constraint trees, so that accurate constraint trees resulted in NJMerge being more accurate than NJ, but inaccurate constraint trees resulted in NJMerge being less accurate than NJ.

4.4.2 How do pipelines using NJMerge compare to ASTRAL-III, SVDquartets, and RAxML?

In this section, we compare the NJMerge pipeline to running Φ_T on the full dataset, where Φ_T is the method used to estimate constraint trees for NJMerge. NJMerge was more accurate when given the AGID matrix (Figure 4.5 and Supplementary Figure S1), so we show results for NJMerge given the AGID matrix and provide results for NJMerge given the log-det distance matrix in the Supplementary Materials.

ASTRAL-III vs. NJMerge: Both NJMerge($\mathcal{T}_{AST}$, D_{AGID}) and NJMerge($\mathcal{T}_{AST}$, D_{LD}) provided substantial running time advantages over ASTRAL-III on datasets with 1 000 species, 1 000 genes, and very high ILS. On 40/40 datasets (20 replicates with exon-like sequences and 20 replicates with intron-like sequences), NJMerge($\mathcal{T}_{AST}$, D_{AGID}) completed in under 300 minutes (approximately 5 hours) on average; this included the time to estimate the dissimilarity matrix, the time to estimate subset trees with ASTRAL-III in a serialized fashion, and the time to combine subset trees using NJMerge (Figure 4.8 and Supplementary Figure S4). ASTRAL-III completed on 17/40 of these dataset but ran for more than 2000 minutes (approximately 33 hours) on average. ASTRAL-III failed to complete within 48 hours (maximum wall-clock time) on the other 23/40 datasets with very high ILS (Table 4.1); however, when the ILS was low/moderate, ASTRAL-III completed (in less than 9 hours on average) on 40/40 datasets. This difference between low/moderate ILS and very high ILS is interesting (see discussion in Section 4.5). In comparison, NJMerge($\mathcal{T}_{AST}$, D_{AGID}) failed on 0 datasets, and NJMerge($\mathcal{T}_{AST}$, D_{LD}) failed on 2 datasets (Table 4.1).

ASTRAL-III and NJMerge($\mathcal{T}_{AST}$, D_{AGID}) achieved similar accuracy with mean species tree error between 0–2% for both intron-like and exon-like datasets (Figure 4.8, Supplementary Figure S4, and Supplementary Table S7). Trends were similar for NJMerge($\mathcal{T}_{AST}$, D_{LD}) except when ILS was very high; under these conditions, the mean species tree error was 2–6% greater for NJMerge($\mathcal{T}_{AST}$, D_{LD}) than for ASTRAL-III (Supplementary Figures S7–S8 and Supplementary Table S8).

NJMerge vs. SVDquartets: The SVDquartets pipeline (as implemented in PAUP*) allows the user to specify whether to run SVDquartets on all (n choose four) possible quartets or on a subset of these quartets. A prior study [180] showed that using all quartets provided the best accuracy, so we ran PAUP* specifying that SVDquartets be run on all (n choose four) possible quartets for datasets with 100 species. This was not possible for datasets with 1 000 species, because the maximum number of quartets allowed by PAUP* was 4.15833×10^{10} at the time of the study. We attempted to run PAUP* on a random subset of quartets (without replacement), but running PAUP* in this fashion resulted in a segmentation fault for all datasets. Thus, SVDquartets failed on 80/80 datasets with 1 000 species and 1 000 genes. In contrast, NJMerge($\mathcal{T}_{SVD}$, D_{AGID}) failed on 0 datasets, and NJMerge($\mathcal{T}_{SVD}$, D_{LD}) failed on 3 datasets (Table 4.1). Thus, NJMerge enabled SVDquartets to complete on datasets with 1 000 species and 1 000 genes.

NJMerge also improved the running time of SVDquartets on datasets with 100 species; for example, SVDquartets completed in 19–81 minutes on average, whereas NJMerge($\mathcal{T}_{SVD}$, D_{AGID}) completed in less than 2 minutes on average for datasets with 100 species and 1 000 genes (Figure 4.9 and Supplementary Figure S5). This running time comparison does not take into account the time needed to estimate gene trees, which required on average 18 minutes using FastTree-2 on datasets with 100 species and 1 000 genes.

NJMerge($\mathcal{T}_{SVD}$, D_{AGID}) typically produced species trees with less error than SVDquartets. The difference between methods was typically small (mean species tree error between 0–2%) when ILS was low/moderate but could be larger than 10% when ILS was very high.

Similar trends were observed for NJMerge($\mathcal{T}_{SVD}$, D_{LD}) (Supplementary Figures S9 and S10).

NJMerge vs. RAxML: Both NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) and NJMerge($\mathcal{T}_{RAX}$, D_{LD}) enabled RAxML to run on datasets with 1 000 species and 1 000 genes. Specifically, RAxML failed to run on 38/40 intron-like datasets and 3/40 exon-like datasets due to "Out of Memory" (OOM) errors (Table 4.1); this difference between intron-like and exon-like datasets is noteworthy (see discussion in Section 4.5). Thus, while NJMerge can fail to return a tree, NJMerge failed less frequently than RAxML when both methods were given the same computational resources. NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) failed on 1 dataset, and NJMerge($\mathcal{T}_{RAX}$, D_{LD}) failed on 2 datasets.

Furthermore, both NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) and NJMerge($\mathcal{T}_{RAX}$, D_{LD}) provided substantial running time advantages over RAxML, often reducing its running time by more than half, even though RAxML was run on the subset trees in a serialized fashion (Figure 4.10 and Supplementary Figure S6). For the exon-like datasets with 1 000 species and 1 000 genes, NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) completed in less than 1 000 minutes (16.6 hours) on average when ILS was low/moderate and in less than 500 minutes (8.3 hours) on average when ILS was very high. In contrast, the final checkpoint was written by RAxML after more than 2250 minutes ($\sim$37.5 hours) on average. Note that the running times for NJMerge do not include the time to estimate gene trees; it took on average 217 minutes (less than 4 hours) to estimate 1 000 gene trees on datasets with 1 000 species using FastTree-2.

The difference in accuracy between RAxML and NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) depended on the model condition, but was typically small. For datasets with low/moderate ILS, RAxML produced species trees with less error (0–3% on average) than NJMerge($\mathcal{T}_{RAX}$, D_{AGID}); however, for datasets with very high ILS, NJMerge($\mathcal{T}_{RAX}$, D_{AGID}) produced species trees with less error (0–4% on average) than RAxML (Figure 4.10 and Supplementary Figure S3). Similar trends were observed for NJMerge($\mathcal{T}_{RAX}$, D_{LD}) (Supplementary Figures S11–S12).

4.5 DISCUSSION

4.5.1 Utility of pipelines using NJMerge

Pipelines using NJMerge can be viewed either as techniques for improving traditional NJ or as techniques for scaling a computationally-intensive base method (denoted Φ_T) to larger datasets. Thus, in order to maximize the utility of NJMerge, users should select a base method that is both more accurate and more computationally intensive than NJ. Our results show that selecting base methods for NJMerge may not be trivial when analyzing multi-locus datasets, because both accuracy and running time were impacted by the level of ILS. For example, ASTRAL-III was very fast when the level of ILS was low/moderate but was substantially slower when the level of ILS was very high. Similarly, when the level of ILS was low/moderate, SVDquartets and RAxML were both more accurate than NJ(D_{AGID}), which is equivalent to NJst; however, SVDquartets and RAxML were less accurate than NJ(D_{AGID}) when the level of ILS was very high. This trend is consistent with results and discussion from Chapter 3. Overall, our results suggest that constraint trees should be estimated using RAxML when the level of ILS is low/moderate and using ASTRAL-III when the level of ILS is very high. Hence, determining the level of ILS in a multi-locus dataset is an important area of future research; a similar conclusion was also drawn in Section 3.5 but regarding the utility of gene filtering. Lastly, NJMerge, when given constraint trees that agreed with the true species tree, was very accurate (less than 2% error on average) even when the level of ILS was very high, suggesting that NJMerge is a promising technique for scaling Bayesian co-estimation methods (e.g., Starbeast2 [43]) and future species tree estimation methods (which are more accurate and more computationally intensive than current methods) to larger datasets.

An important issue is that NJMerge can fail to return a tree. In our experiments, NJMerge failed on 11/2560 test cases from running NJMerge on 320 datasets with two different types of distance matrices and four different types of constraint trees (Table 4.1). Therefore, NJMerge failed on fewer datasets than ASTRAL-III, SVDquartets, or RAxML when all methods were given the same computational resources: a single compute node with 64 GB of memory, 16 cores, and a maximum wall-clock time of 48 hours. It is worth noting that NJMerge was run within the divide-and-conquer pipeline shown in Figure 4.4, so subsets of species were created by decomposing the NJ tree (blue dashed lines); our results on the failure rate of NJMerge likely do not generalize to arbitrary inputs.

Impact of dissimilarity matrix on NJ: Our results showed that on average NJ(D_{AGID}) was as accurate or more accurate than NJ(D_{LD}). There was a clear difference between these two methods on datasets with 100 species, 1 000 intron-like genes, and low/moderate ILS; specifically NJ(D_{AGID}) produced trees with less than 5% error on average, whereas NJ(D_{LD}) produced trees with greater than 12% error on average). However, on the exact same model condition but with 1 000 species, NJ(D_{AGID}) and NJ(D_{LD}) produced trees with similar levels of accuracy (mean species tree error was 11%). Interestingly, for low/moderate ILS datasets only, the median branch length varied with the number of species, so that overall, the 1 000-species datasets had shorter internal branches than the 100-species datasets (Supplementary Table S3). Branch length and other factors that limit the accuracy of NJ(D_{LD}) in the context of gene tree estimation may also apply in the context of species tree estimation. Even so, NJ(D_{LD}) was more accurate than either SVDquartets or RAxML when the level of ILS was very high, providing support for Allman *et al.*'s statement, "The simplicity and speed of distance-based inference suggests log-det based methods should serve as benchmarks for judging more elaborate and computationally-intensive species trees inference methods" [41].

Impact of ILS and sequence type on ASTRAL-III: Our results showed that ASTRAL-III was much faster on the low/moderate ILS datasets than on the very high ILS datasets. This finding makes sense in light of ASTRAL-III's algorithm design. ASTRAL-III operates by searching for an optimal solution to its search problem within a constrained search space that is defined by the set Σ of bipartitions in the estimated gene trees, and in particular, ASTRAL-III's running time scales with $|\Sigma|^{1.726}$ [158]. The set of gene trees will become more heterogeneous for higher levels of ILS, and thus, the size of Σ will increase, as every gene tree could be different when the level of ILS is very high. In addition, GTEE can also increase the size of Σ, explaining why ASTRAL-III failed to complete on exon datasets more often than on intron datasets (Table 4.1 Supplementary Table S2).

Impact of sequence type on RAxML: Our results showed that RAxML failed on more intron-like datasets than exon-like datasets. This finding makes sense in light of RAxML's implementation. RAxML uses redundancy in site patterns to store the input MSA compactly, so memory scales with the number of unique site patterns. Introns are less conserved that exons and thus have more unique site patterns, explaining why RAxML required more memory to analyze intron-like datasets.

4.5.2 Statistical consistency of pipelines using NJMerge

The probability that NJMerge fails goes to zero as the number of true gene trees generated under the MSC model goes to infinity. This is a consequence of Corollary 4.2, which relies on the choice of heuristics for determining whether or not to accept a siblinghood proposal. Indeed, it is easy to think of other heuristics that prevent NJMerge from failing but do not have the guarantee of correctness (Theorem 4.2) and therefore do *not* have the guarantee of statistical consistency (Corollary 4.2). Finally, our proof of statistical consistency under the MSC model requires that the number of true gene trees goes to infinity, which is equivalent to requiring that both the number of gene trees and the number of sites per gene go to infinity. Roch *et al.* [125] recently showed that many (if not all) gene tree summary methods can be statistically inconsistent under the MSC model when the number of sites per gene is bounded; these theoretical results apply to divide-and-conquer pipelines using NJMerge.

4.6 CONCLUSIONS

In this chapter, we introduced a divide-and-conquer approach to phylogeny estimation that divides a set of species into pairwise disjoint subsets, builds a tree on each subset of species using a base method, and then merges the subset trees together using a dissimilarity matrix. For the merger step, we presented a new method, called NJMerge and proved that some divide-and-conquer pipelines using NJMerge are statistically consistent under the GTR model and the MSC model. We then evaluated pipelines using NJMerge in the context of species tree estimation, specifically using simulated multi-locus datasets with two levels of ILS and up to 1 000 species and 1 000 genes. We found that pipelines using NJMerge provided several benefits to large-scale species tree estimation; specifically, under some model conditions, pipelines using NJMerge improved the accuracy of traditional NJ and substantially reduced the running time of three popular species tree methods (ASTRAL-III, SVDquartets, and concatenation analysis using RAxML) without sacrificing accuracy (see Section 4.5 for details as the results depended on the level of ILS).

These theoretical and empirical results suggest that NJMerge is a promising approach; however, there are still open challenges. First, NJMerge can fail to return a tree. While this did not occur very often in our simulation study (NJMerge failed on only 11 out of 2 560 test cases), the potential for failure is still an undesirable attribute for a method. Second, NJMerge may be more scalable than other methods, but its worst-case running time is quite terrible $O(n^4 \log^2 n)$ and its quadratic storage requirement is untenable for very large numbers of species. Lastly, although NJMerge could be naively implemented for a distributed-memory system, its performance would suffer from the global communication (all-to-one and one-to-all) required to coordinate each of the n siblinghood joins. We address these challenges in Chapter 5 and discuss further opportunities for future research.

This section contains the six plots presented in Section 4.4 Results.

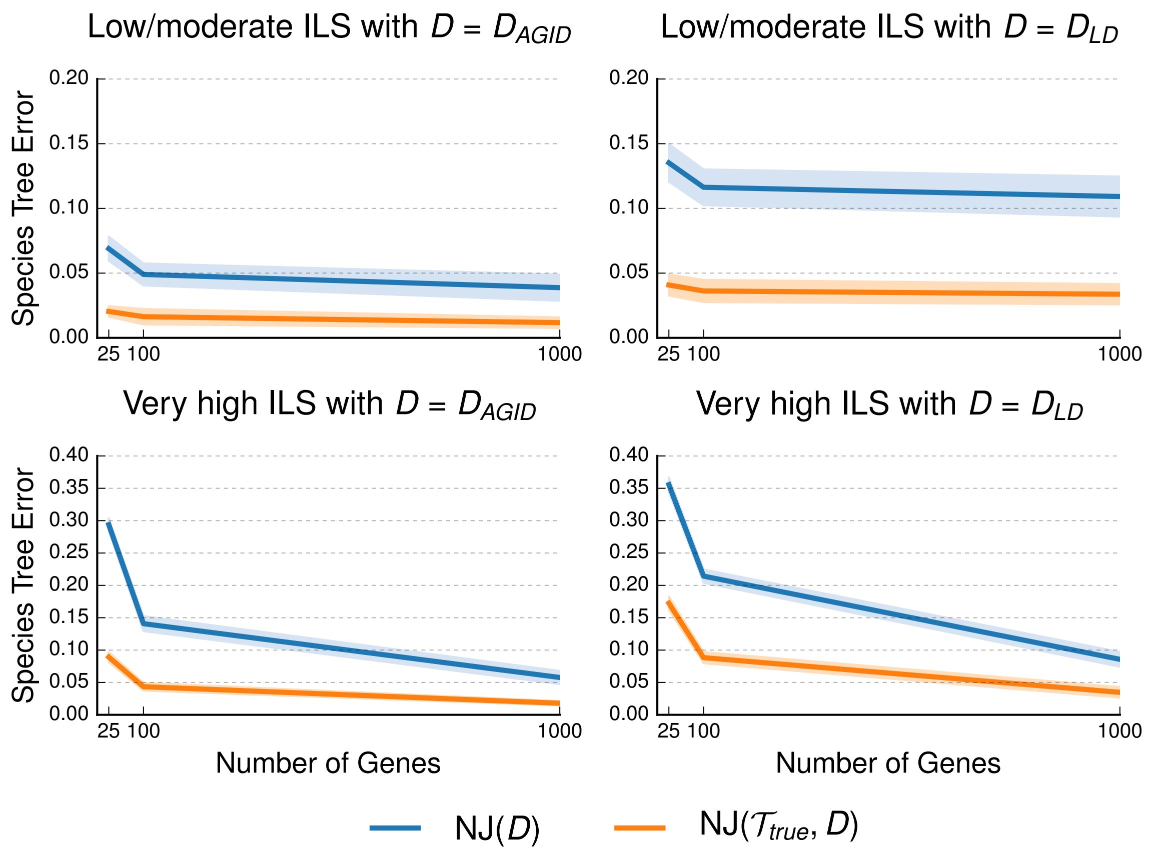


Figure 4.5: **Impact of estimated dissimilarity matrix on NJMerge.** NJ and NJMerge were benchmarked on distance matrices estimated using two different metrics; in addition, NJMerge was given constraint trees that agreed with the true species tree (see Section 4.3 for notation). Datasets had 100 species, 25 to 1 000 intron-like genes, and two levels of ILS. Species tree error is the RF error rate. Lines represent the average over replicate datasets, and filled regions indicate the standard error.

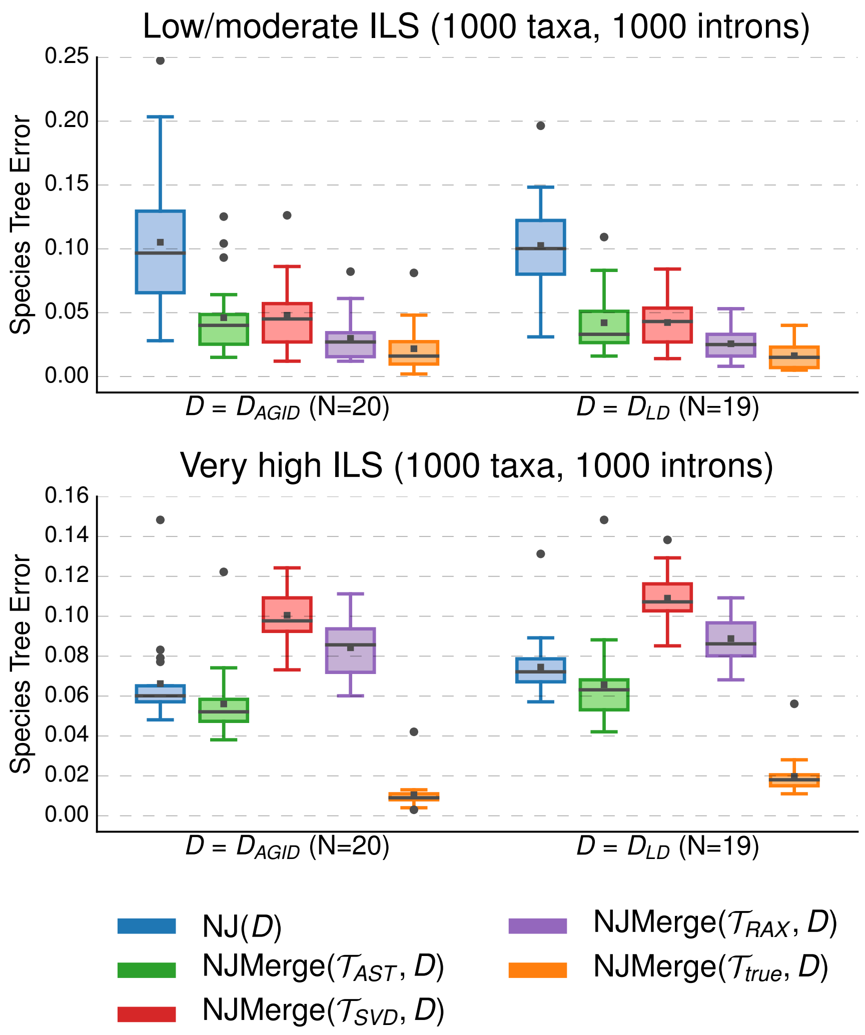


Figure 4.6: **Impact of estimated constraint trees on NJMerge.** NJ and NJMerge were benchmarked on distance matrices estimated using two different metrics; in addition, NJMerge was given constraint trees estimated using four different techniques (see Section 4.3 for notation). Datasets had 1 000 species, 1 000 intron-like genes, and two levels of ILS. Species tree error is the RF error rate. Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value).

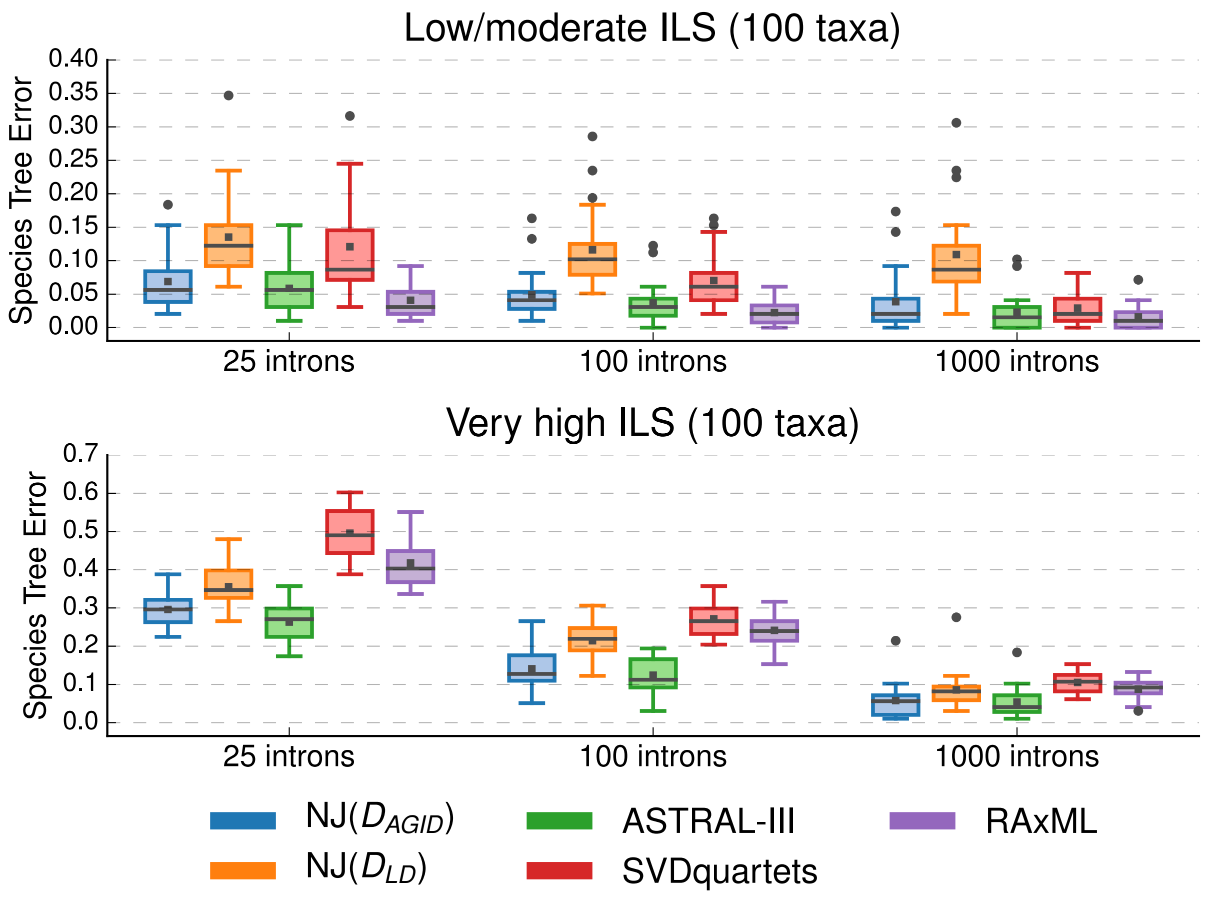


Figure 4.7: **Comparison of species tree methods.** NJ was benchmarked on distance matrices estimated using two different metrics (see Section 4.3 for notation). Datasets had 100 species, 25 to 1 000 intron-like genes, and two levels of ILS. Species tree estimation error is the RF error rate. Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value).

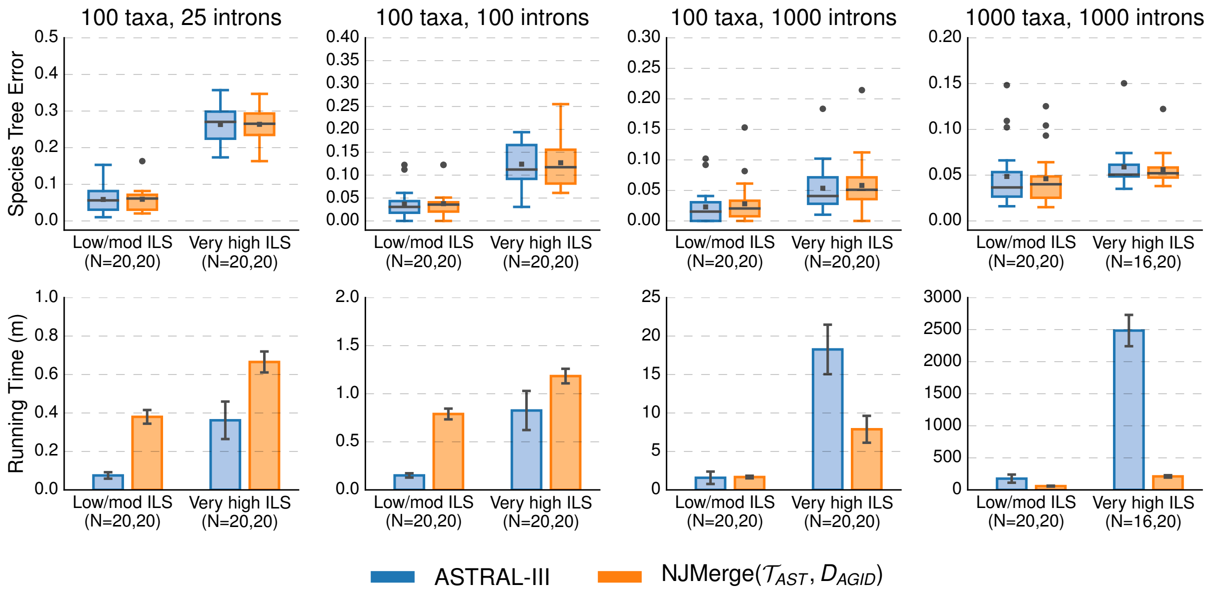


Figure 4.8: **ASTRAL-III vs. NJMerge given ASTRAL-III constraint trees and AGID matrix.** Subplots in the top row show species tree error (RF error rate). Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). Subplots in the bottom row show running time (in minutes); bars represent means and error bars represent standard deviations across replicate datasets. NJMerge running times are for computing the subset trees in a serialized fashion (Equation 4.2). The numbers of replicates on which the methods completed is shown on the x-axis; for example, $N = X, Y$ indicates that ASTRAL-III completed on X out of 20 replicates and that NJMerge($\mathcal{T}_{AST}, D_{AGID}$) completed on Y out of 20 replicates. ASTRAL-III did not complete within the maximum wall-clock time of 48 hours on 4/40 intron-like datasets with 1 000 species (taxa) and very high ILS.

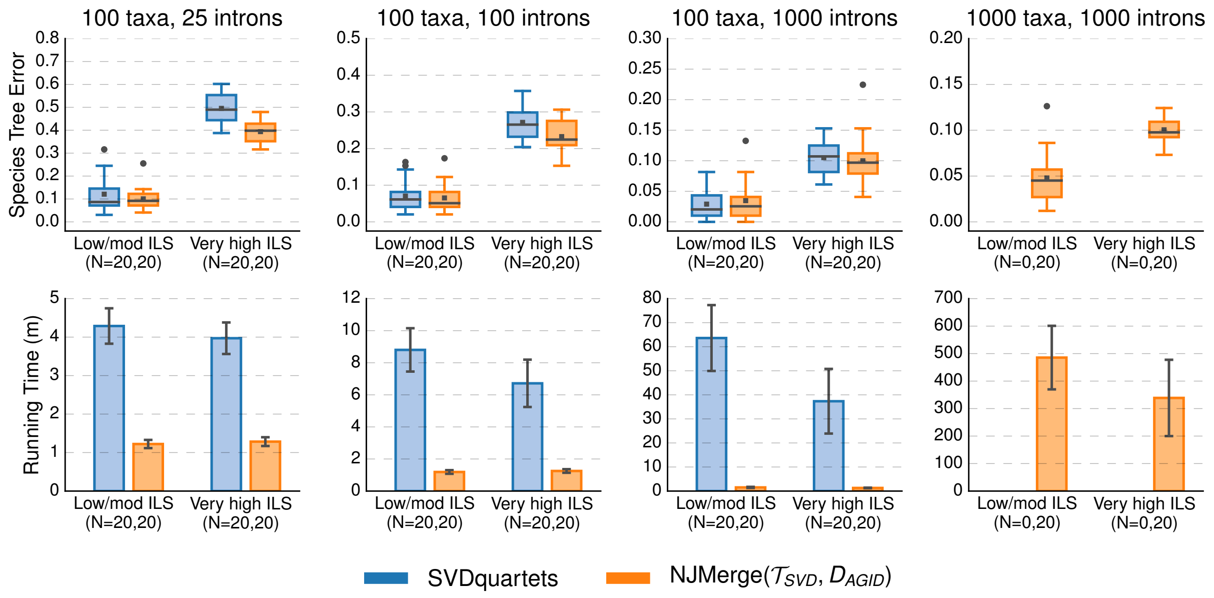


Figure 4.9: **SVDquartets vs. NJMerge given SVDquartet constraint trees and AGID matrix.** Subplots in the top row show species tree error (RF error rate). Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). Subplots in the bottom row show running time (in minutes); bars represent means and error bars represent standard deviations across replicate datasets. NJMerge running times are for computing the subset trees in a serialized fashion (Equation 4.2). The numbers of replicates on which the methods completed is shown on the x-axis; for example, $N = X, Y$ indicates that SVDquartets completed on X out of 20 replicates and that NJMerge($\mathcal{T}_{SVD}, D_{AGID}$) completed on Y out of 20 replicates. SVDquartets did not run any datasets with 1 000 species (taxa) due to segmentation faults.

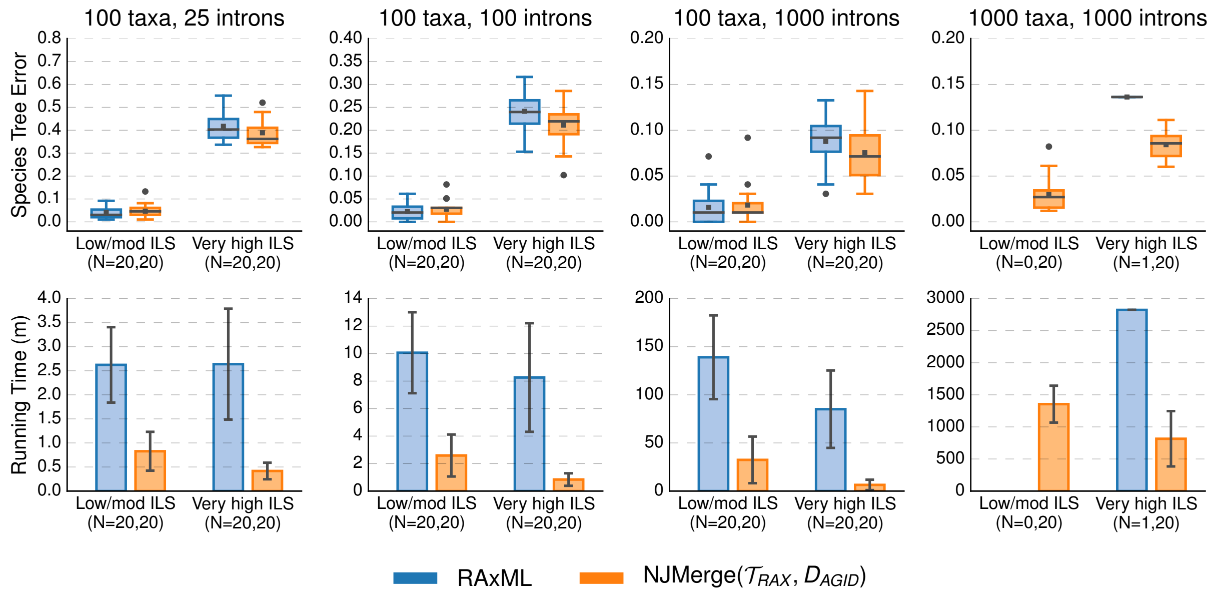


Figure 4.10: **RAxML vs. NJMerge given RAxML constraint trees and and AGID matrix.** Subplots in the top row show species tree error (RF error rate). Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). Subplots in the bottom row show running time (in minutes); bars represent means and error bars represent standard deviations across replicate datasets. NJMerge running times are for computing the subset trees in a serialized fashion (Equation 4.2). The numbers of replicates on which the methods completed is shown on the x-axis; for example, $N = X, Y$ indicates that RAxML completed on X out of 20 replicates and that NJMerge($\mathcal{T}_{RAX}, D_{AGID}$) completed on Y out of 20 replicates. RAxML was only able to run on 1/40 intron-like datasets with 1 000 species (taxa) due to "Out of Memory" errors.

4.8 TABLE

This section contains the table presented in Section 4.4 Results.

Table 4.1: **Failures.** The number of datasets on which methods failed is indicated by model condition. ASTRAL-III failed due to running beyond the maximum wall clock time of 48 hours, SVDquartets failed due to segmentation faults, RAxML failed due to running out of memory, and NJMerge failed due to being unable to find a legal siblinghood. Note that NJMerge is described by the input set $\mathcal{T}$ of constraint trees and input dissimilarity matrix D (see Section 4.3 for notation).

# of Species	# of Genes	ILS Level	Sequence Type	Method	# of Failures (out of 20)
100	25	very high	exon	NJMerge($\mathcal{T}_{true}$, D_{LD})	1
100	25	very high	exon	NJMerge($\mathcal{T}_{RAX}$, D_{AGID})	1
100	25	very high	intron	NJMerge($\mathcal{T}_{true}$,D_{AGID})	1
1 000	1 000	low/moderate	exon	SVDquartets	20
1 000	1 000	low/moderate	exon	RAxML	3
1 000	1 000	low/moderate	intron	NJMerge($\mathcal{T}_{AST}$, D_{LD})	1
1 000	1 000	low/moderate	intron	SVDquartets	20
1 000	1 000	low/moderate	intron	RAxML	20
1 000	1 000	very high	exon	ASTRAL-III	19
1 000	1 000	very high	exon	NJMerge($\mathcal{T}_{true}$, D_{LD})	1
1 000	1 000	very high	exon	NJMerge($\mathcal{T}_{AST}$, D_{LD})	1
1 000	1 000	very high	exon	NJMerge($\mathcal{T}_{SVD}$, D_{LD})	2
1 000	1 000	very high	exon	NJMerge($\mathcal{T}_{RAX}$, D_{LD})	2
1 000	1 000	very high	exon	SVDquartets	20
1 000	1 000	very high	intron	ASTRAL-III	4
1 000	1 000	very high	intron	NJMerge($\mathcal{T}_{SVD}$, D_{LD})	1
1 000	1 000	very high	intron	SVDquartets	20
1 000	1 000	very high	intron	RAxML	19

4.9 ALGORITHMS

This section contains the three algorithms presented in Section 4.2 Approach.

Algorithm 4.2: NJMerge.

Input : Set $\mathcal{T} = \{T_i\}_{i=1}^{k}$ of unrooted phylogenetic trees such that $S(T_i) \cap S(T_j) = \emptyset \ \forall \ i \neq j$ and an $n \times n$ dissimilarity matrix D on label set $S = \bigcup_{i=1}^{k} S(T_i)$

Output: A (possibly refined) compatibility supertree for $\mathcal{T}$ that is fully resolved

```
Function NJMerge($\mathcal{T}$, $D$):
    $I \leftarrow \{1, 2, \ldots, n\}$; $S \leftarrow$ list version of $S$
    Relabel $T_i \in \mathcal{T}$, so a leaf labeled $s$ is now labeled $i$ if row $i$ in $D$ is labeled $s$
    for $i \in I$ do $r[i] \leftarrow \sum_{j \in I} D[i,j]$

    while $|I| > 3$ do

        Step 1: Build $Q$, so that $Q[i,j] \leftarrow (|I|-2)D[i,j] - r[i] - r[j] \ \forall \ i,j \in I$, and
        simultaneously sort $Q$ from smallest to largest, producing the vector sorted.

        Step 2: Select next sibling pair $(x,y)$ to join.
        for $i \in \{1, 2, \ldots, (|I|^2 - |I|)/2\}$ do
            $(x,y) \leftarrow sorted[i]$
            $pass \leftarrow$ RunCompatiblityHeuristicAndUpdateConstraints($\mathcal{T}$, $x$, $y$)
            if $pass$ then break
        if not pass then return $fail$

        Step 3a: Update $D$.
        $\vec{dx} \leftarrow [0]_n$; $dxy \leftarrow D[x,y]$
        for $i \in I \setminus \{x,y\}$ do
            $dx[i] \leftarrow D[x,i]$
            $D[x,i] \leftarrow \frac{1}{2}\big(dx[i] + D[y,i] - dxy\big)$; $D[i,x] \leftarrow D[x,i]$

        Step 3b: Update $\vec{r}$.
        for $i \in I \setminus \{x,y\}$ do
            $r[i] \leftarrow r[i] - dx[i] - D[y,i] + D[x,i]$
        $r[x] \leftarrow \sum_{i \in I \setminus \{x,y\}} D[x,i]$

        Step 3c: Update indices and labels.
        $I \leftarrow I \setminus \{y\}$
        $S[x] \leftarrow$ '(' + S[x] + ','+ S[y] + ')'

    $T \leftarrow [NULL]_3$; $j \leftarrow 1$
    for $i \in I$ do
        $T[j] \leftarrow L[i]$; $j \leftarrow j + 1$

    return '(' + T[1] + ','+ T[2] + ',' + T[3] + ');'
```

Algorithm 4.3: RunCompatiblityHeuristicAndUpdateConstraints. We assume that the representation of each constraint tree T uses $O(n)$ space and includes an n-vector $L(T)$ of leaf nodes, so that the element at position i is either the leaf with label i or $NULL$ if there is no leaf with label i. This allows leaf nodes to be accessed by their labels in constant time.

Input : Set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of unrooted phylogenetic trees and labels $x, y \in S = \bigcup_{i=1}^{k} S(T_i)$
Output: *true* if the test passes, and $false$ otherwise

```
Function RunCompatiblityHeuristicAndUpdateConstraints($\mathcal{T}$, $x$, $y$):
    for $i \in \{1, 2, \ldots, k\}$ do
        $L \leftarrow L(T_i)$
        if $L[x] \neq NULL$ and $L[y] \neq NULL$ then
            if not IsSiblingPair($T_i$, $x$, $y$) then return $fail$
    $\mathcal{P} = \emptyset$; $update \leftarrow [0]_k$
    for $i \in \{1, 2, \ldots, k\}$ do
        $update[i] \leftarrow$ UpdateTree($T_i$, $x$, $y$)
        if $update[i] \neq 0$ then
            $\mathcal{P} \leftarrow \mathcal{P} \cup T_i$
    if IsCompatibleHeuristic($\mathcal{P}$) then return $true$
    for $i \in \{1, 2, \ldots, k\}$ do ReverseUpdateTree($T_i$, $x$, $y$, $update[i]$)
    return $fail$

Function UpdateTree($T$, $x$, $y$):
    $L \leftarrow L(T)$
    if $L[x] \neq NULL$ and $L[y] \neq NULL$ then
        $update \leftarrow 1$; PruneLeaf($T$, $y$)
    else if $L[y] \neq NULL$ then
        $update \leftarrow 2$; RelabelLeaf($T$, $y$, $x$);
    else if $L[x] \neq NULL$ then
        $update \leftarrow 3$
    return $update$

Function ReverseUpdateTree($T$, $x$, $y$, $update$):
    if $update = 1$ then
        AddSibling($T$, $x$, $y$)
    else if $update = 2$ then
        RelabelLeaf($T$, $x$, $y$)
```

Algorithm 4.4: IsCompatibleHeuristic. We assume that the representation of each constraint tree T uses $O(n)$ space and includes an n-vector $L(T)$ of leaf nodes, so that the element at position i is either the leaf with label i or $NULL$ if there is no leaf with label i. This allows leaf nodes to be accessed by their labels in constant time. Note that this is a naive implementation.

Input : Set $\mathcal{P} = \{T_1, T_2, \ldots, T_p\}$ of unrooted phylogenetic trees
Output: $true$ if the test passes, and $false$ otherwise

```
Function IsCompatibleHeuristic($\mathcal{P}$):
    for $i \in \{1, 2, \ldots, p-1\}$ do
        for $j \in \{i+1, \ldots, p\}$ do
            $L_i \leftarrow L(T_i)$; $L_j \leftarrow L(T_j)$
            $R \leftarrow \emptyset$
            for $s \in S(T_i) \cap S(T_j)$ do
                if $L_i[s] \neq NULL$ and $L_j[s] \neq NULL$ then
                    $R \leftarrow R \cup \{s\}$
            $P'_i \leftarrow$ CopyTree($P_i$); RestrictTree($P'_i$, $R$)
            $P'_j \leftarrow$ CopyTree($P_j$); RestrictTree($P'_j$, $R$)
            $rf \leftarrow$ ComputeRF($P'_i$, $P'_j$)
            DeleteTree($P'_i$); DeleteTree($P'_j$)
            if $rf > 0$ then
                return $false$
    return $true$

Function RestrictTree($T$, $R$):
    for $l \in S(T) \setminus R$ do
        PruneLeaf($T$, $l$)
```

CHAPTER 5: DIVIDE-AND-CONQUER PIPELINES WITH TREEMERGE

This chapter contains material previously published in "TreeMerge: a new method for improving the scalability of species tree estimation methods" [236], which was joint work with T. Warnow. Software commands necessary to reproduce this study are freely available on the Illinois Data Bank: doi.org/10.13012/B2IDB-9570561_V1. TreeMerge *is freely available on Github: github.com/ekmolloy/treemerge. Note that plots and tables appear at the end of this chapter in Sections 5.7 and 5.8, respectively.*

5.1 INTRODUCTION

In Chapter 4, we proposed an alternative approach to divide-and-conquer that operates by (i) dividing the species set into pairwise disjoint subsets of a predetermined maximum size, (ii) estimating a tree on each subset, (iii) computing any auxiliary data required for merging subset trees, and then (iv) using the auxiliary data to merge the subset trees together into a compatibility supertree (Definition 2.7). For the final step, we presented NJMerge, a disjoint tree merger (DTM) method that uses a dissimilarity matrix to build a (possibly refined) compatibility supertree for the subset trees. Despite its promising theoretical and empirical results, NJMerge has two major issues that limit its utility in practice. First, NJMerge can fail to return a tree, and second, the worst-case running time of NJMerge is $O(n^4 k)$, where the input has n species (leaf labels) divided across k leaf-label-disjoint trees.

In this chapter, we address the limitations of NJMerge by presenting new methods, all of which are guaranteed to return a compatibility supertree when given (fully resolved) leaf-label-disjoint trees as input. The first method, NJMerge-2, is a minor modification to NJMerge; it has the same theoretical properties as NJMerge except that it does not fail. The second method, *TreeMerge*, seeks to address the scalability issue by merging subset trees in two phases: a *local merge phase* and a *global merge phase*. In the local merge phase, a highly accurate but computationally intensive method is used to merge leaf-label-disjoint trees in an embarrassingly parallel fashion. In the global merge phase, the trees computed during the local merge phase are then combined via a reduction. Within the divide-and-conquer framework, TreeMerge runs in $O(n \log n)$ time or in $O(n^2)$ time if the reduction is serialized. This running time analysis assumes that the mergers during the global phase are performed using a linear-time technique.

As we will show, the linear-time merge operation constrains the space of allowed solutions, which in turn places some additional (but achievable) requirements for divide-and-conquer pipelines using TreeMerge to be statistically consistent (as compared to those using NJMerge or NJMerge-2). Nevertheless, the results of our simulation study show that species trees estimated using TreeMerge can be quite accurate, comparing favorably to those estimated by other DTM-based divide-and-conquer pipelines or traditional species tree estimation methods (ASTRAL-III and CA-ML using RAxML). Like NJMerge, TreeMerge enables the dominant species tree estimation methods to scale to larger datasets without sacrificing accuracy. Unlike NJMerge, TreeMerge is guaranteed to return a compatibility supertree, is much faster (in terms of its worst-case running time), and is more easily parallelizable, making it a notable advance for large-scale phylogeny estimation.

5.2 APPROACH

We now present the NJMerge-2 and TreeMerge algorithms focusing on worst-case running time, correctness, and statistical consistency within divide-and-conquer pipelines.

5.2.1 NJMerge-2

NJMerge-2 is a simple extension to NJMerge so that it is guaranteed to return a compatibility supertree for the input constraint (subset) trees. Recall that NJMerge modifies NJ by imposing a set of topological constraints on the output tree. For each siblinghood proposal, NJMerge updates the constraint trees based on the proposed siblinghood and then tests the compatibility of the updated constraint trees; if the test passes, NJMerge accepts the siblinghood proposal. Because determining the compatibility of a set of k unrooted trees on overlapping leaf label sets is NP-complete [71], NJMerge uses a heuristic that can fail for $k > 2$ trees. Therefore, by running NJMerge on two constraint trees at a time, it will never fail; this observation is the basis of NJMerge-2 (Algorithm 5.1).

Theorem 5.1. Let $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ be a set of unrooted phylogenetic trees such that $S(T_i) \cap S(T_j) = \emptyset$ for all $i \neq j$, and let D be an $n \times n$ dissimilarity matrix on label set $S = \cup_{i=1}^{k} S(T_i)$. Then, NJMerge-2 applied to input $(\mathcal{T}, D)$ returns a (possibly refined) compatibility supertree for $\mathcal{T}$ in $O(n^4 k)$ time; the parallel version of NJMerge-2 runs in $O(n^4)$ time. Furthermore, if D is a nearly additive matrix for a fully resolved tree T^* on label set S (Definition 2.11) and if T^* is compatible with T_i for all $i \in \{1, \ldots, k\}$ (Definition 2.1), then NJMerge-2 returns T^*.

Proof. Returns a compatibility supertree: It is easy to see that the constraint trees remain leaf-label-disjoint and thus compatible at each iteration, and because the heuristics used by NJMerge correctly determine compatibility for two trees, NJMerge applied to the input $\big(\{t, t'\}, D|_{S(t) \cup S(t')}\big)$ is guaranteed to return a (possibly refined) compatibility supertree (Theorem 4.1). By induction on the number of constraint trees, NJMerge-2 returns a (possibly refined) compatibility supertree for $\mathcal{T}$.

Correctness: If, in addition, the trees in $\mathcal{T}$ are compatible with T^* and the dissimilarity matrix D is nearly additive for T^*, then for any pair of trees $t, t' \in \mathcal{T}$, NJMerge applied to the input $\big(\{t, t'\}, D|_{S(t) \cup S(t')}\big)$ returns $T^*|_{S(t) \cup S(t')}$ (Theorem 4.2). Therefore, the set $\mathcal{T}$ remains compatible with T^* during the iterative process. By induction on the number of constraint trees, NJMerge-2 returns T^*.

Running time: For the running time analysis, we make the simplifying assumption that each of the k trees in $\mathcal{T}$ has exactly n/k leaves. At iteration w, for any two trees $t, t' \in \mathcal{T}$, $|S(t) \cup S(t')| \leq (w+1)n/k$, so the worst-case running time of NJMerge given any pair of trees is $O(w^4 n^4 / k^4)$ (Theorem 4.1). A total of $k - 1$ iterations are required, so the running time of NJMerge-2 scales with $(n^4/k^4) \sum_{w=2}^{k} w^4$. Because $\sum_{w=2}^{k} w^4$ is $O(k^5)$, the worst-case running time of NJMerge-2 is $O(n^4 k)$. Although Algorithm 5.1 runs NJMerge-2 on pairs of trees in serial, we could consider a parallel version of NJMerge-2. Assuming that k is a power of two for simplicity, at iteration w, we run NJMerge on each of the $k/(2^w)$ pairs of trees in parallel, where for any pair of trees $t, t' \in \mathcal{T}$, $|S(t) \cup S(t')| = 2^w n/k$. A total of $p = \log_2(k) - 1$ iterations are required, so the parallel running time of NJMerge scales with $(n^4/k^4) \sum_{w=2}^{p} 2^{4w}$ and thus is $O(n^4)$. This is expected as the running time of NJMerge-2 is dominated by the time to run NJMerge in the final iteration. QED.

Algorithm 5.1: NJMerge-2.

Input : Set $\mathcal{T} = \{T_i\}_{i=1}^{k}$ of unrooted phylogenetic tree such that $S(T_i) \cap S(T_j) = \emptyset$ for all $i \neq j$ and an $n \times n$ dissimilarity matrix D on label set $S_{i=1}^{k} \bigcup S_i$

Output: A (possibly refined) compatibility supertree for $\mathcal{T}$ that is fully resolved

```
Function NJMergeTwoSerial($\mathcal{T}$, $D$, $S$):
    while $|\mathcal{T}| > 1$ do

        $t, t' \leftarrow$ An arbitrary pair of trees in $\mathcal{T}$
        $R \leftarrow [S(t) \cup S(t')]$
        $D|_R \leftarrow$ CopyMatrixRestricted($D$, $R$)
        $T \leftarrow$ NJMerge($\{t, t'\}$, $D|_R$, $R$)
        DeleteMatrix($D|_R$)
        $\mathcal{T} \leftarrow \big(\mathcal{T} \setminus \{t, t'\}\big) \cup \{T\}$

    return $T$
```

We do not specify an order for merging pairs of constraint trees in Algorithm 5.1. While the order does not impact the theoretical guarantees of NJMerge-2, it likely impacts performance (accuracy) in practice. Consider a collection of constraint trees that are edge separable for a caterpillar tree T with a very large *evolutionary diameter* (the longest evolutionary distance between any pair of leaves in a phylogenetic tree). If, in the first iteration, we merge two constraint trees on opposite sides of the caterpillar tree, the output tree may contain incorrect bipartitions due to long branch attraction; these incorrect bipartitions will be propagated through subsequent iterations. Hence, it may be beneficial to perform mergers in an order that respects locality by selecting pairs of trees $t, t' \in \mathcal{T}$ based on the evolutionary distance between set $S(t)$ and set $S(t')$ in D. Exploiting locality can also be useful for improving computationally efficiency, as we will show.

5.2.2 TreeMerge

We now introduce TreeMerge, the first DTM method that combines an embarrassingly parallel local merge phase with a global merge phase. As shown in Algorithm 5.2, these two phases are enabled through the use of a *merge guide tree* given as part of the input. The generic TreeMerge algorithm also requires auxiliary data in order to build compatibility supertrees during both the local and the global merge phases. The exact details of the algorithm can be implemented in a variety of ways; we present two examples: TreeMerge-fast (which was previously mentioned) and TreeMerge-slow.

- **TreeMerge Input**:
  - Set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of unrooted, fully resolved phylogenetic trees such that $S(T_i) \cap S(T_j) = \emptyset$ for all $i \neq j$
  - Set $\mathcal{A}$ of auxiliary data, including
    - Merge guide tree $\mathcal{G}$ with nodes bijectively labeled by elements of the set $\{1, 2, \ldots, k\}$ such that node i corresponds to tree $T_i \in \mathcal{T}$
    - Data that can be used to merge trees, for example
      - *TreeMerge-fast:* Set $\mathcal{D} = \{D^{i,j} : (i,j) \in E(\mathcal{G})\}$ of dissimilarity matrices
      - *TreeMerge-slow:* Dissimilarity matrix D on label set $S = \bigcup_{i=1}^{k} S(T_i)$
- **TreeMerge Output**: Compatibility supertree for $\mathcal{T}$

We now describe the local and global phases for TreeMerge-slow and TreeMerge-fast.

Algorithm 5.2: Generic TreeMerge Algorithm.

```
Input  : Set $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ of unrooted, fully resolved phylogenetic trees with constraint
         tree $T_i$ on label set $S_i$ such that $S_i \cap S_j = \emptyset$ for all $i \neq j$, a set $\mathcal{A}$ of auxiliary data, which
         includes a tree $\mathcal{G}$, called the merge guide tree, with nodes bijectively labeled by elements
         of the set $\{1, 2, \ldots, k\}$ so that node $i$ corresponds to tree $T_i \in \mathcal{T}$.
Output: Compatibilitysupertree for $\mathcal{T}$

Function GenericTreeMerge($\mathcal{T}$, $\mathcal{A}$):

    Local merge phase: For each edge $(i, j) \in E(\mathcal{G})$, build a compatibility supertree $T_{i,j}$ for
    $\{T_i, T_j\}$ using some auxiliary data in $\mathcal{A}$

    Global merge phase:
    $(x, y) \leftarrow$ An arbitrary edge in $E(\mathcal{G})$
    $T \leftarrow T_{x,y}$
    $\mathcal{G} \leftarrow \mathcal{G}$ rooted at edge $(x, y)$
    for $(i, j) \in$ PreOrderEdgeTraversal($\mathcal{G}$) do
        $T \leftarrow$ Compatibility supertree for $\{T, T_{i,j}\}$ built using their backbone tree $T|_{S(T) \cap S(T_{i,j})}$
          and possibly some auxiliary data in $\mathcal{A}$

    return $T$
```

Local merge phase: During the local merge phase, mergers are performed on pairs of trees that are local, meaning that they are connected by edges in the merge guide tree $\mathcal{G}$. For every edge $(i, j) \in E(\mathcal{G})$, both TreeMerge-slow and TreeMerge-fast build a compatibility supertree $T_{i,j}$ by running NJMerge on the input $(\{T_i, T_j\}, D^{i,j})$. Afterward, TreeMerge-fast uses $D^{i,j}$ to fit branch lengths to $T_{i,j}$ via the least squares approach proposed by Bryant and Waddell [237]. Note that TreeMerge-fast takes $D^{i,j}$ as input, whereas TreeMerge-slow creates $D^{i,j}$ by restricting D to label set $S(T_i) \cup S(T_j)$.

Global merge phase: During the global merge phase, mergers are performed following a pre-order edge traversal of the merge guide tree $\mathcal{G}$, combining the trees computed during the local merge phase into a single tree on label set S (note that this can also be implemented as a reduction; see Figure 5.1 for an example). The merge technique is related to the Strict Consensus Merger [221, 226].

We describe this merge technique in the context of merging two trees $T_{i,j}$ and $T_{j,k}$ that are compatibility supertrees for $\{T_i, T_j\} \subset \mathcal{T}$ and $\{T_j, T_k\} \subset \mathcal{T}$, respectively. The input trees in $\mathcal{T}$ are fully resolved, so $T_{i,j}$ and $T_{j,k}$ induce a common tree topology T_j when restricted to their shared label set; we refer to T_j as the *backbone tree*. Each edge e in $E(T_j)$ maps to a path $p_{i,j}(e)$ in $T_{i,j}$ and a path $p_{j,k}(e)$ in $T_{j,k}$. When $p_{i,j}(e)$ or $p_{j,k}(e)$ have length greater than one, the internal nodes on those paths define subtrees that need to be inserted onto edge e in the backbone tree T_j. Let $V_{i,j}(e)$ denote the set of subtrees in $T_{i,j}$ that need to be attached to edge e, and similarly, let $V_{j,k}(e)$ denote the set of subtrees in $T_{j,k}$ that need to be attached to edge e. It is easy to see that adding subtrees in the set $V_{i,j}(e) \cup V_{j,k}(e)$ to edge e in an arbitrary order and then repeating this process for all edges in $E(T_j)$, produces a (possibly refined) compatibility supertree for $\{T_{i,j}, T_{j,k}\}$ and thus for $\{T_i, T_j, T_k\}$. If $T_{i,j}$ and $T_{j,k}$ never contribute subtrees to the same edge, $T_{i,j,k}$ is the unique compatibility supertree for $\{T_i, T_j, T_k\}$; otherwise, $T_{i,j,k}$ is a refined compatibility supertree.

When $T_{i,j}$ and $T_{j,k}$ both contribute subtrees to the same edge (referred to as a *collision*), there are multiple ways to merge the two trees while maintaining compatibility (Figure 5.2). Picking the best resolution cannot be done using tree topologies of $T_{i,j}$ and $T_{j,k}$ alone. TreeMerge-fast and TreeMerge-slow differ with respect to how they resolve collisions.

TreeMerge-fast: TreeMerge-fast resolves edge collisions by using the branch lengths fitted to $T_{i,j}$ and $T_{j,k}$ during the local merge phase. Suppose that $T_{i,j}$ and $T_{j,k}$ each contribute one or more subtrees to an edge e in their shared backbone tree T_j. Then, edge e has two different lengths: the length given by path $p_{i,j}(e)$ in $T_{i,j}$ and the length given by path $p_{j,k}(e)$ in $T_{j,k}$. TreeMerge-fast rescales these paths so that they have same length, producing an order for subtrees in $V_{i,j}(e) \cup V_{j,k}(e)$ to be added to edge e. This approach does not allow subtrees contributed by $T_{i,j}$ and $T_{j,k}$ to blend together (Figure 5.2), limiting the potential solutions that can be returned by TreeMerge-fast. It is not possible for TreeMerge-fast to recover the correct tree for some subset decomposition and merge guide tree pairs (Figure 5.3).

TreeMerge-slow: TreeMerge-slow resolves edge collisions between two trees $T_{i,j}$ and $T_{j,k}$ using the shared backbone tree T_j as follows. Let $e = (X', Y') \in E(T_j)$ be an edge involved in a collision, and select two labels $X, Y \in S(T_j)$ corresponding to leaves on opposite sides of e. We define a constraint tree t by restricting $T_{i,j}$ to leaf label set $\{X, Y\} \cup \{S(v) : v \in V_{i,j}(e)\}$, and we define another constraint tree t' by restricting $T_{j,k}$ to leaf label set $\{X, Y\} \cup \{S(v) : v \in V_{j,k}(e)\}$. Because t and t' have only leaves X and Y in common, they are compatible; therefore, a compatibility supertree for $\{t, t'\}$ can be computed in polynomial time by running NJMerge on the input $\left(\{t, t'\}, D_{S(t) \cup S(t')}\right)$ (Theorem 4.1). The compatibility supertree for $\{t, t'\}$ can be inserted into T_j by attaching leaf X at node X' and leaf Y at node Y'. This technique allows t and t' to blend (Figure 5.2).

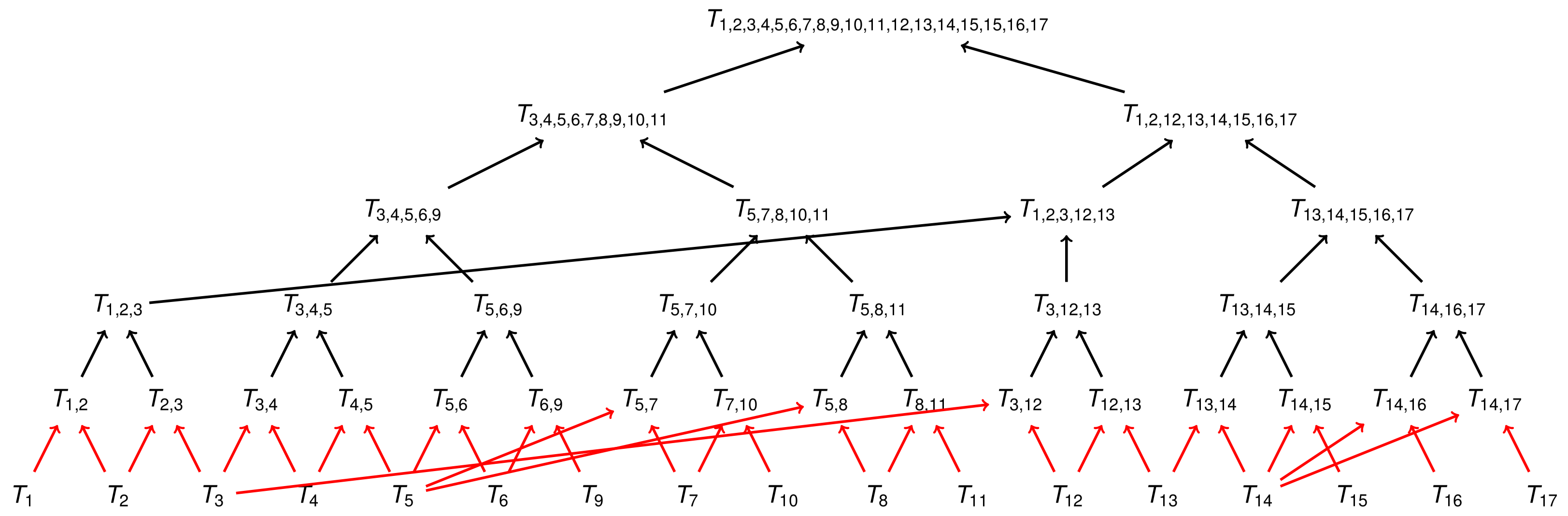


Figure 5.1: **Parallel Reduction for the Generic TreeMerge.** It is well known that a reduction can be performed on a tree. This reduction is for subset trees $\mathcal{T} = \{T_1, T_2, \ldots, T_{17}\}$ and merge guide tree $\mathcal{G}$ given by edge set: $\{(1,2),(2,3),(3,4),(4,5),(5,6),(6,9),(5,7),(7,10),(5,8),(8,11),(3,12),(12,13),(13,14),(14,15),(14,16),(14,17)\}$. Red arrows indicate that two trees (e.g., T_{14} and T_{17}) are given as input to a DTM method (e.g., using NJMerge). Black arrows indicate that two trees (e.g., $T_{14,17}$ and $T_{14,16}$) are merged via their shared backbone tree (e.g., using branch lengths as in TreeMerge-fast). The local merge phase is one step (red arrows), and the global merge phase takes $\log_2(16) = 4$ steps (black arrows). Mergers at each of these steps can be performed in an embarrassingly parallel fashion.

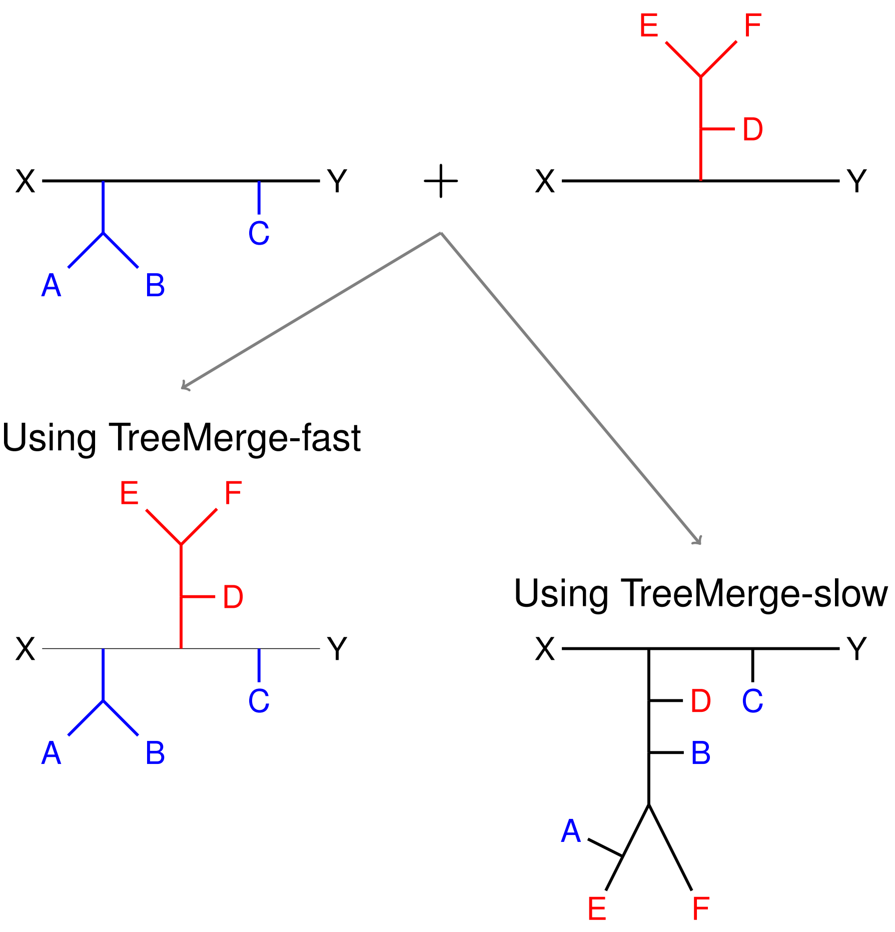
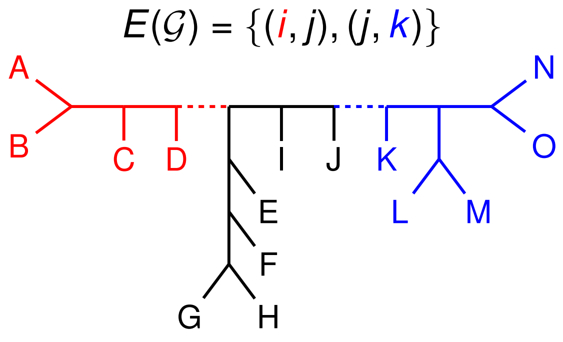
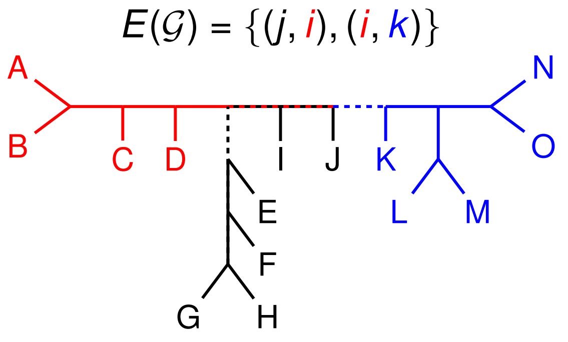


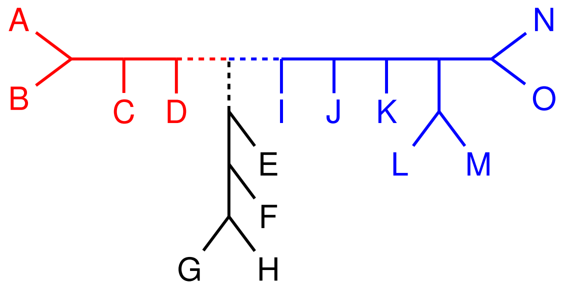
Figure 5.2: **TreeMerge-slow vs. TreeMerge-fast.** TreeMerge-slow and TreeMerge-fast differ with respect to how they resolve edge collisions. Consider the case where two compatible trees $T_{i,j}$ and $T_{j,k}$ are involved in a collision on edge $e = (X', Y')$ in the backbone tree T_j, using the notation from Section 5.2.2. Select two leaves X and Y on opposite sides of the edge e in T_j. We define a constraint tree t (upper left corner) by restricting $T_{i,j}$ to leaf label set $\{x, y\} \cup \{S(v) : v \in V_{i,j}(e)\}$, so the subtrees in $V_{i,j}(e)$ are given by the Newick strings: (A, B) and (C). We define another constraint tree t' (upper right corner) by restricting $T_{j,k}$ to leaf label set $\{x, y\} \cup \{S(v) : v \in V_{j,k}(e)\}$, so the one subtree in $V_{i,k}(e)$ is given by the Newick string: $(D, (E, F))$. The goal is to produce a compatibility supertree t^* for $\{t, t'\}$ and then insert t into the backbone tree T_j by attaching leaf X at internal node X' and attaching leaf Y at internal node Y'. TreeMerge-fast builds t^* by rescaling branch lengths (if necessary) so that the paths from X to Y in t and t' have the same length (as shown here); t is then defined by superimposing these paths. When resolving collisions, TreeMerge-fast does not allow *blending*, because the subtrees in the set $V_{i,j}(e) \cup V_{j,k}(e) \cup \{X, Y\}$ must be edge separable for t^*. In contrast, TreeMerge-slow builds t^* by running NJMerge on the input $\left(\{t, t'\}, D|_{S(t) \cup S(t')}\right)$; this enables blending.



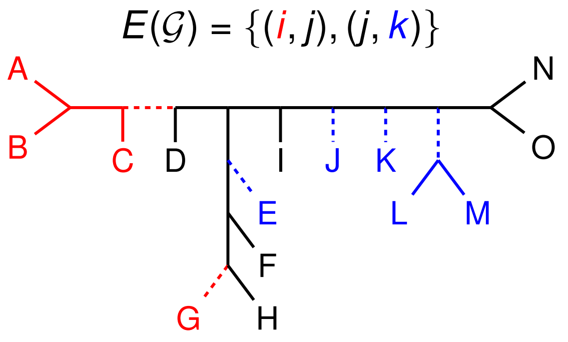
(a) No collisions $\{T_{i,j}, T_{j,k}\}$



(b) Collisions $\{T_{j,i}, T_{i,k}\}$



(c) Collision for all $\mathcal{G}$



(d) No collisions $\{T_{i,j}, T_{j,k}\}$

Figure 5.3: **Impact of collisions on the correctness of TreeMerge-fast.** For each subfigure, the true tree T is shown. Colors indicate how the leaves of T have been decomposed into subsets: subset i, j, and k are shown in red, black, and blue, respectively. Suppose that subset trees T_i, T_j, or T_k are correct and thus can be created by restricting T to the red, black, or blue leaves, respectively. The merge guide tree $\mathcal{G}$ indicates how the TreeMerge algorithm proceeds. For example, in subfigure (a), $\mathcal{G}$ indicates that $\{T_i, T_j\}$ and $\{T_j, T_k\}$ are merged during the local phase. Again, suppose that the resulting trees are correct; therefore, $T_{i,j}$ can be created by restricting T to the red and black leaves, and $T_{j,k}$ can be created by restricting T to the black and blue leaves. During the global phase, TreeMerge-fast merges $T_{i,j}$ and $T_{j,k}$ via the backbone tree T_j. Subfigure (a) shows no collisions, so TreeMerge-fast returns the correct tree. Subfigure (b) shows a collision on the edge with bipartition $A, B, C|D$ in T_j, so TreeMerge-fast returns the incorrect tree; for example, TreeMerge-fast could return the tree given by Newick string: $((A, B), C)$, $(((((G, H), F), E), (I, J)), (K, ((L, M), (O, N)))),$ $D)$. Note that subfigure (b) shows the same subset decomposition as subfigure (a), but $\mathcal{G}$ is different. Subfigure (c) shows a collision occurring regardless of $\mathcal{G}$, and TreeMerge returns the incorrect tree for this decomposition regardless of the $\mathcal{G}$; for example, TreeMerge-fast run on $\{T_{i,j}, T_{i,k}\}$ could return $(((A, B), C),$ $(((G, H), F), E),$ $(I, (J, (K, ((L, M), (O, N))))),$ $D)$. Subfigure (d) shows no collisions, so TreeMerge-fast returns the correct tree. Unlike in the other subfigures, the subset trees $\{T_i, T_j, T_j\}$ are not edge separable for T.

We now provide some theoretical guarantees for TreeMerge-fast and TreeMerge-slow.

Theorem 5.2. Suppose that the input $(\mathcal{T}, \mathcal{A})$ has n leaf labels divided across k leaf-label-disjoint trees in $\mathcal{T}$. Then, TreeMerge-fast applied to $(\mathcal{T}, \mathcal{A})$ returns a compatibility supertree for $\mathcal{T}$ in $O(nk + n^2/k + n^4/k^3)$ time if using the serial version and $O(n \log_2 k + n^2/k^2 + n^4/k^4)$ time if using the parallel version. TreeMerge-fast requires $O(n^2/k)$ storage.

In contrast, TreeMerge-slow applied to $(\mathcal{T}, \mathcal{A})$ returns a compatibility supertree for $\mathcal{T}$ in $O(n^4 k)$ time if using the serial version and $O(n^4)$ time if using the parallel version. TreeMerge-slow requires $O(n^2)$ storage.

Proof. The proof that both TreeMerge-slow and TreeMerge-fast return a compatibility supertree follows from the technique for building compatibility supertrees (in the local merge phase) returning a compatibility supertree when given two leaf-disjoint trees as input and from the techniques for resolving collisions (in the global merge phase) returning a compatibility supertree when given two trees that agree on their shared leaf label set.

For the running time analysis, we make the simplifying assumptions that each tree in $\mathcal{T}$ has exactly n/k leaves and that $k-1$ is a power of two.

Local merge phase: In the local merge phase, both TreeMerge-slow and TreeMerge-fast run NJMerge on input $(\{T_i, T_j\}, D^{i,j})$ for each of the $k-1$ edges in the merge guide tree $\mathcal{G}$. All input constraint trees have n/k leaves, so the local merge phase uses $O(n^2/k)$ storage. The running time of the local merge phase is $O(n^4/k^3)$ if mergers are performed in serial and $O(n^4/k^4)$ if mergers are performed in parallel.

Extended local merge phase for TreeMerge-fast: TreeMerge-fast extends the local merge phase by computing branch lengths for each of the $k-1$ trees using the quadratic time and space algorithm from [237]. The running time of the extended local merge phase is then $O(n^2/k)$ if performed in serial and $O(n^2/k^2)$ if performed in parallel.

TreeMerge-fast and TreeMerge-slow implement the global merge phase differently; we discuss these methods separately using the notation from Algorithm 5.2.

Global merge phase for TreeMerge-fast: At iteration w, TreeMerge-fast merges trees $T_{i,j}$ and T using their shared backbone tree t_B and branch lengths. This is just a special case of the algorithm proposed by Bansal [238] for Optimal Tree Completion under the Robinson-Foulds (RF) Distance. This algorithm scales linearly with the number of leaf labels in the output tree, so $|S(T_{i,j}) \cup S(T)| = (w+2)n/k$. A total of $k-2$ iterations are required, so the running time scales with $(n/k)\sum_{w=3}^{k} w$. Because $\sum_{w=3}^{p} w$ is $O(p^2)$, the worst-case running time of TreeMerge-fast is $O(kn)$. It is possible to perform this global merge phase in parallel via a reduction on the merge guide tree $\mathcal{G}$. We merge trees using branch lengths in $\log_2(k-1)$ iterations and each iteration is $O(n)$; therefore, in the parallelized global merge phase, TreeMerge-fast uses $O(n \log_2 k)$ time.

Global merge phase for TreeMerge-slow: At iteration w, TreeMerge-slow merges trees $T_{i,j}$ and T using their shared backbone tree t_B, where $|S(T_{i,j})| = 2n/k$, $|S(T)| = (w+1)n/k$, and $|S(t_B)| = n/k$. There are $(w+1)n/k$ possible leaves that could be contributed to the $n/k - 3$ edges in t_B. In the worst case analysis, $(w+1)n/k$ leaves are contributed to a single edge in t_B, so NJMerge uses $O(w^4n^4/k^4)$ time and $O(w^2n^2/k^2)$ storage. A total of $k-2$ iterations are required, so the running time scales with $(n^4/k^4)\sum_{w=3}^{k} w^4$. Because $\sum_{w=3}^{p} w^4$ is $O(p^5)$, the worst-case running time of TreeMerge-slow is $O(kn^4)$. In the serialized global merge phase, TreeMerge-slow uses $O(n^4k)$ time and uses $O(n^2)$ storage. Notably, the worst-case analysis is effectively the same as the analysis for NJMerge-2 (essentially just specifying the order of the mergers). It is possible to perform the global merge phase in parallel via a reduction on the merge guide tree $\mathcal{G}$; the worst-case analysis is effectively the parallel version of NJMerge-2, so TreeMerge-slow requires $O(n^4)$ time.

Overall, TreeMerge-slow uses $O(n^2)$ storage and TreeMerge-fast uses $O(n^2/k)$ storage. For the serialized versions, TreeMerge-slow runs in $O(n^4k)$ time and TreeMerge-fast runs in $O(nk + n^2/k + n^4/k^3)$ time. For the parallelized versions, TreeMerge-slow runs in $O(n^4)$ time and TreeMerge-fast runs in $O(n\log_2 k + n^2/k^2 + n^4/k^4)$ time. QED.

Note that while the worst-case running time of TreeMerge-slow is the same as the worst-case running time of NJMerge-2, these two methods can have very different running times in practice, for example when there are very few collisions. In fact, when there are zero collisions, TreeMerge-slow and TreeMerge-fast have the same running time. That being said, it is possible for large collisions to occur; for example, if TreeMerge-slow was run on the case presented in Figure 5.3b, it would run NJMerge on two red leaves (D and either A, B, or C), all black leaves, and all blue leaves, so basically the entire tree!

Theorem 5.3. If a phylogenetic tree T^* is compatible with every tree in $\mathcal{T}$, and D is nearly additive for T^*. Then, TreeMerge-slow returns T^*.

Proof. Let T_x and T_y be two trees in $\mathcal{T}$. NJMerge applied to $\big(\{T_x, T_y\}, D|_{S(T_x)\cup S(T_y)}\big)$ returns $T^*|_{S(T_x)\cup S(T_y)}$ by Theorem 4.2. As TreeMerge-slow performs all its mergers using NJMerge, the result follows by induction on the number of mergers. QED.

Theorem 5.4. If a phylogenetic tree T^* is compatible with every tree in $\mathcal{T}$, and D is nearly additive for T^*. Then, TreeMerge-fast is guaranteed to return T^* provided that there are no collisions during the global merge phase.

Proof. If there are no collisions during the global merge phase, TreeMerge-fast performs all of its mergers with NJMerge; the result follows by induction on the number of mergers. QED.

5.2.3 Divide-and-conquer Pipelines for Phylogeny Estimation and Statistical Consistency

We now provide some theoretical guarantees for divide-and-conquer pipelines that use either NJMerge-2, TreeMerge-slow, or TreeMerge-fast to combine subset trees; see Algorithm 4.1.

Corollary 5.1. When run within the divide-and-conquer framework proposed in Algorithm 4.1, TreeMerge-fast runs in $O(n^2)$ time if using the serial version and $O(n \log_2 n)$ time if using the parallel version. In contrast, the parallel versions of NJMerge-2 and TreeMerge-slow run in $O(n^4)$ time, and the serial versions of NJMerge, NJMerge-2, and TreeMerge-slow run in $O(n^5)$ time.

Proof. In the divide-and-conquer framework, the input dataset of n species is divided into k pairwise disjoint subsets of bounded size, so $c = n/k$ and therefore $k = n/c = O(n)$. Then, the result follows from updating the equations for worst-case running time given in Table 5.1. QED.

Corollary 5.2. NJMerge-2 and TreeMerge-slow can be used in a gene tree estimation pipeline that is statistically consistent under the Generalized Time Reversible (GTR) model, and they can be used in a species tree estimation pipeline that is statistically consistent under the Multi-Species Coalescent (MSC) model.

The proof, which is similar to the proofs of Corollaries 4.1 and 4.2, follows easily from Theorems 5.1 and 5.3. To summarize, divide-and-conquer pipelines using either NJMerge, NJMerge-2, or TreeMerge-slow are statistically consistent when the method Φ_D used to estimate distances between pairs of species and the method Φ_T used to estimate constraint trees are statistically consistent under the model of interest. These proofs of statistical consistency do not depend on the subset decomposition, the order that subsets are merged (for NJMerge-2), or the merge guide tree (for TreeMerge-slow). Because TreeMerge-fast can fail to combine two trees correctly when collisions occur (Figure 5.3), divide-and-conquer pipelines using TreeMerge-fast are statistically consistent under a model if the probability of collisions goes to zero, as the number of samples generated under the model goes to infinity.

Theorem 5.5. Consider the following pipeline:

- Estimate a starting tree T_0 using method Φ_0. Decompose the starting tree T_0 into pairwise disjoint subsets $S_1, S_2, \ldots, S_k$ by removing a set E_0 of edges from T_0 such that every pair of edges in E_0 is separated by at least two edges in T_0 (e.g., Figure 5.3a). Label node v in T_0 by i if v is on a path between two leaves that are both in S_i. Define the merge guide tree $\mathcal{G}$ on vertex set $\{1, 2, \ldots, k\}$ by making i and j adjacent in $\mathcal{G}$ if and only if there is an edge in T_0 whose endpoints are labeled by i and j.

- Build a dissimilarity matrix $D^{i,j}$ by running method Φ_D to estimate distances between all pairs of species in the set $S_i \cup S_j$; repeat for every edge $(i, j) \in \mathcal{G}$ to produce a set $\mathcal{D}$ of dissimilarity matrices.

- Estimate a tree T_i on subset S_i using method Φ_T; repeat for all $i \in \{1, 2, \ldots, k\}$ to produce a set $\mathcal{T}$ of constraint trees.

Suppose that Φ_0, Φ_D, and Φ_T are statistically consistent under some model of evolution. Then, as the number of independent samples generated under the model goes to infinity, the phylogenetic tree computed using TreeMerge-fast on input $(\mathcal{T}, \{\mathcal{D}, \mathcal{G}\})$ will converge to the true (model) tree T^*. In other words, this divide-and-conquer pipeline using TreeMerge-fast is statistically consistent.

Proof. As the number of samples increases, each estimated dissimilarity matrix $D^{i,j} \in \mathcal{D}$ will converge to a matrix that is additive for $T^*|_{S(T_i) \cup S(T_j)}$, each estimated constraint tree $T_i \in \mathcal{T}$ will converge to $T^*|_{S_i}$, and the estimated starting tree T_0 will converge to T^*. Hence, for a large enough number of samples, the following occurs with high probability. (i) The deletion of edges in E_0 from the T_0 produces subsets such that the labeling of T_0 described above assigns each internal node with a unique label in the set $\{1, 2, \ldots, k\}$; in other words, the labeling is convex on T_0 and thus on T^*. (ii) If i and j are adjacent in the merge guide tree $\mathcal{G}$, then $T^*|_{S_i \cup S_j}$ can be created by connecting T_i and T_j with an edge. It is easy to see that under these conditions there will no collisions with high probability, so by Theorem 5.4, TreeMerge-fast will return T^* with high probability. QED.

5.3 PERFORMANCE STUDY

We present the results of using TreeMerge-fast to estimate species trees on large multi-locus datasets simulated for the NJMerge study (Section 4.3.1). Recall that these datasets had 1 000 species and 1 000 genes and were characterized by two levels of incomplete lineage sorting (ILS) (low/moderate and very high) and two data types (exons and introns). In this study, we focus our attention on TreeMerge-fast comparing it to NJMerge and NJMerge-2. Note that NJMerge-2 and TreeMerge-slow have the same theoretical properties for worst-case running time and statistical consistency (Table 5.1), but NJMerge-2 is much simpler to implement.

Table 5.1: **Theoretical properties of NJMerge, NJMerge-2, TreeMerge-slow, and TreeMerge-fast.**

- "Can Fail?" means that the method can, on some inputs, fail to return a tree due to algorithmic issues (rather than limited computational resources).
- "Consistent?" means that the method is statistically consistent under the GTR and MSC models when used within the divide-and-conquer pipelines described in Corollaries 4.1 and 4.2, respectively. Yes* indicates that a modified divide-and-conquer pipeline is needed to ensure statistical consistency for TreeMerge-fast (Theorem 5.5).
- "Running time" and "storage" are for worst-case analysis.
- "D&C runtime" is the running time of the method when used within the divide-and-conquer framework (Algorithm 4.1).

Note that TreeMerge-slow has the same worst-case running time and other properties as NJMerge-2; however, it may be faster in practice, for example when there are very few collisions.

	Can Fail?	Consistent?	Running time	Storage	D&C Runtime
NJMerge	Yes	Yes	$O(n^4 k)$	$O(n^2)$	$O(n^5)$
NJMerge-2	No	Yes	$O(n^4 k)$	$O(n^2)$	$O(n^5)$
Parallel NJMerge-2	No	Yes	$O(n^4)$	$O(n^2)$	$O(n^4)$
TreeMerge-fast	No	Yes*	$O(nk + n^2/k + n^4/k^3)$	$O(n^2/k)$	$O(n^2)$
Parallel TreeMerge-fast	No	Yes*	$O(n \log_2(k) + n^2/k^2 + n^4/k^4)$	$O(n^2/k)$	$O(n \log n)$

We also perform a study similar to the NJMerge study (Section 4.3), comparing TreeMerge-fast to two of the dominant species tree estimation methods: ASTRAL-III and CA-ML using RAxML. Recall that this type of study compares running a method Φ_T within a divide-and-conquer pipeline to estimate subset trees versus running Φ_T *de novo* (i.e., on the full dataset). All methods are evaluated with respect to their algorithmic failure rate (which is only relevant to NJMerge), computational failure rate (failure to complete due to insufficient computational resources), (empirical) running time, and species tree accuracy.

It is worth noting that the divide-and-conquer pipelines tested in these experiments (which were performed prior to writing Theorem 5.5) do not guarantee that the requirements for TreeMerge-fast to be statistically consistent were met.

5.3.1 Divide-and-Conquer Pipelines

Divide-and-conquer pipelines require the user to specify several inputs (Algorithm 4.1). We created two different pipelines: the ASTRAL-III pipeline used the estimated gene trees as input, and the RAxML pipeline used the concatenated alignment as input.

Divide-and-conquer pipeline using ASTRAL-III:

- *Dissimilarity matrix:* AGID matrix computed using ASTRID version 1.4 given estimated gene trees as input
- *Starting tree:* NJ tree computed using FastME version 2.1.5 on the AGID distance matrix (i.e., the NJst tree)
- *Constraint trees:* Species trees computed using ASTRAL version 5.6.1 (i.e., ASTRAL-III) given the estimated gene trees restricted to a specific subset of species as input

Divide-and-conquer pipeline using RAxML:

- *Dissimilarity matrix:* Matrix of log-det distances computed using PAUP* version 4a163 given the concatenated alignment as input
- *Starting tree:* Greedy maximum parsimony tree based on a random taxon addition order computed using RAxML version 8.2.12 (with SSE3 and pthreads) given the concatenated alignment as input
- *Constraint trees:* Species trees computed under the GTR+GAMMA model of evolution using RAxML version 8.2.12 (with SSE3 and pthreads) given the concatenated alignment restricted to a specific subset of species as input

Subset decomposition: For both pipelines, pairwise disjoint subsets of species were created by applying the centroid edge decomposition to the estimated starting tree T_0. Note that the centroid edge decomposition does not prevent edges incident to the same node in T_0 from being deleted, so the requirements for TreeMerge-fast to be statistically consistent are not guaranteed to be satisfied by this pipeline (Theorem 5.5).

Running TreeMerge-fast: TreeMerge-fast requires as input a merge guide tree $\mathcal{G}$, which is a tree with vertices bijectively labeled by the trees in $\mathcal{T}$. We used the starting tree T_0 to construct $\mathcal{G}$ as follows. First, we randomly selected one leaf from each tree in $\mathcal{T}$ and deleted all other leaves from T_0, suppressing internal nodes of degree 2; this produced a tree T_0' that had one leaf for every tree in $\mathcal{T}$. Second, we built a complete graph $\mathcal{G}_0$ with nodes labeled by the trees in $\mathcal{T}$ and edges (T_i, T_j) weighted by the path distance between leaves corresponding to T_i and T_j in T_0'. Third, we computed a minimum spanning tree $\mathcal{G}$ on $\mathcal{G}_0$ using Kruskal's algorithm [239]. This approach for producing $\mathcal{G}$ does not guarantee that requirements for TreeMerge-fast to be statistically consistent are met (Theorem 5.5). To make a fair comparison between TreeMerge-fast and NJMerge-2, the order that subset trees were merged by NJMerge-2 was equivalent to the order that subset trees were merged by TreeMerge-fast during the global merge phase.

TreeMerge-fast also requires branch lengths to be estimated on the trees produced during the local merge phase; this was accomplished by using PAUP* version 4a163 to fit least squares positive branch lengths to each of these trees using the same dissimilarity matrix from the local merge phase.

5.3.2 Evaluation

Methods were evaluated in terms of species tree error, measured as the RF error rate (Equation 2.3), and running time, measured as the wall-clock time recorded in seconds. All computational experiments were run on the Blue Waters supercomputer, as described in Section 4.3.4. We compared the time to run method Φ_T *de novo* (i.e., on the full dataset) to the time to run the entire divide-and-conquer pipeline; this was approximated as

$$time\big(\Phi_D(X)\big) + \sum_{T \in \mathcal{T}} time\big(\Phi_T(X|_{S(T)})\big) + time\big(\Phi_M(\mathcal{T}, \mathcal{A})\big) \tag{5.1}$$

where Φ_M denotes the method to merge constraint trees using auxiliary information $\mathcal{A}$; see Algorithm 4.1 for other notation. Equation 5.1 does not include the time to estimate starting trees, as required a few minutes compared to hundreds or thousands of minutes.

5.4 RESULTS

5.4.1 How does TreeMerge compare to NJMerge and NJMerge-2?

We first evaluate the impact of running either NJMerge, NJMerge-2, and TreeMerge-fast within two divide-and-conquer pipelines (one using ASTRAL-III on subsets and one using RAxML on subsets). First, we discuss the cases where NJMerge failed to return a tree; these failures were algorithmic rather than computational. When ASTRAL-III was used to construct subset trees, NJMerge failed to return a tree on 0/80 datasets; however, when RAxML was used to construct subset trees, NJMerge failed to return a tree on 6/80 datasets (i.e., the failure rate was 7.5%). For the same analyses in the NJMerge study (Section 4.3), only 2/80 datasets resulted in failures (i.e., the failure rate was 2.5%). The only difference between the analyses here and the ones in Section 4.3 is the starting tree; in this study, we used the greedy (randomized) parsimony tree from RAxML as the starting tree, whereas in the previous study, we used the NJ tree computed from the log-det distance matrix. This finding suggests that the choice of starting tree may be a factor in whether or not NJMerge fails. In contrast, NJMerge-2 and TreeMerge-fast completed on all datasets within the allowed time using the allowed memory.

On the replicate datasets for which all methods returned a tree, we compared NJMerge, NJMerge-2, and TreeMerge-fast in terms of species tree accuracy. We found that NJMerge-2 produced trees with the same average error as those produced by NJMerge and that TreeMerge-fast produced trees with at most 1% greater error on average than those produced by NJMerge (Table 5.2).

That TreeMerge-fast had similar performance compared to NJMerge and NJMerge-2 is noteworthy, as the ASTRAL-III pipeline guarantees statistical consistency for NJMerge and for NJMerge-2 but does *not* for TreeMerge-fast. In particular, the centroid edge decomposition and the construction of the merge guide tree $\mathcal{G}$ do not meet the requirements described in Theorem 5.5. This suggests the possibility that TreeMerge-fast might be even more accurate when used within pipelines that provide a guarantee of statistical consistency, which we note in Section 5.6 as a topic for future research.

5.4.2 What is the impact of using TreeMerge-fast on ASTRAL-III and RAxML?

We now evaluate TreeMerge-fast within two divide-and-conquer pipelines (one using ASTRAL-III on subsets and one using RAxML on subsets) compared to running these methods *de novo*.

Failure rate: In our experiments, ASTRAL-III and RAxML failed to complete analyses on many datasets though for different reasons. ASTRAL-III failed to complete its analyses within 48 hours on 19/40 exon datasets and 4/40 intron datasets (i.e., the combined failure rate was 29%); note that all of these failures occurred on datasets with very high ILS. RAxML reported Out Of Memory (OOM) errors on 3/40 exon datasets and 39/40 intron datasets (i.e., the combined failure rate was 53%). In contrast, when run within divide-and-conquer pipelines using TreeMerge-fast, all analyses with ASTRAL-III and RAxML completed. Thus, TreeMerge-fast enabled both ASTRAL-III and RAxML to complete analyses on large datasets when given only 64 GB of memory and 48 hours wall-clock time.

Notably, the failure rate for ASTRAL-III and RAxML depended on the model condition. RAxML failed (due to running out of memory) on more intron datasets than exon datasets. Exon-like sequences, which evolve more slowly than intron-like sequences, had fewer distinct alignment patterns and thus could be more effectively compressed. When the alignments could not be effectively compressed (as was the case for the intron datasets), RAxML was more likely to run out of memory. A distributed-memory version of RAxML, called ExaML, can be used to estimate trees when RAxML runs out of memory, provided that users have access to a distributed-memory system; in our study, we explicitly limited all methods to a single compute node, assuming users had limited computational resources. ASTRAL-III failed (due to running longer than 48 hours) on datasets with very high ILS. High ILS datasets are characterized by true gene trees that are topologically very different from each other and from the true species tree. The running time of ASTRAL-III depends on the degree of gene tree heterogeneity [158], explaining why ASTRAL-III failed to complete within 48 hours on many of the high ILS datasets.

On the replicate datasets for which ASTRAL-III or RAxML completed, we compared running the methods *de novo* to running them within the divide-and-conquer pipeline based on TreeMerge-fast.

Running time: We found that running ASTRAL-III or RAxML within the divide-and-conquer pipeline (using TreeMerge-fast) reduced running time, often dramatically (Figures 5.4 and 5.5). For example, when the level of ILS was very high, running ASTRAL-III within the divide-and-conquer pipeline reduced the total running time from 42 hours to 4 hours on average. Similarly, for exon datasets, running RAxML within the divide-and-conquer pipeline reduced the total running time from 43 hours to 11 hours on average. The time required to estimate subset trees using ASTRAL-III or RAxML typically dominated the total running time of the divide-and-conquer pipeline (with one exception: using ASTRAL-III to estimate subset trees on low ILS datasets). Merging subset trees together with TreeMerge-fast was relatively fast, requiring no more than 46 minutes, which was between 3–11% of the average time required to estimate subset trees using RAxML (Table 5.3). We prototyped TreeMerge-fast (without parallelism) in Python using dendropy [240], so an optimized implementation (with parallelism) would produce even better results.

Species tree error: We found that whether ASTRAL-III or RAxML were run *de novo* or within the divide-and-conquer pipeline (using TreeMerge-fast) had little impact on species tree accuracy, with error rates differing by at most 1% on average (Figures 5.4 and 5.5). These effects were most pronounced for datasets with very high ILS; specifically, ASTRAL-III was more accurate (by 1% on average) when run *de novo*, whereas RAxML was less accurate (by 1% on average) when run *de novo*.

5.5 DISCUSSION

5.5.1 Computational Complexity, Accuracy, and Statistical Consistency

The proof of statistical consistency for divide-and-conquer pipelines using TreeMerge-fast depends on the techniques used to decompose the species set into subsets and to build the merge guide tree. In Theorem 5.5, we show how this can be accomplished by estimating a starting tree on the full set of species using a statistically consistent method (e.g., a distance method). Notably, divide-and-conquer pipelines using NJMerge (Algorithm 4.1) also compute the subset decomposition based on a starting tree.

While the statistical consistency of divide-and-conquer pipelines using NJMerge does not depend on estimating a starting tree, there is no reason not to perform the subset decomposition in this way. NJMerge already requires an $n \times n$ dissimilarity matrix and runs $O(n^5)$ within the divide-and-conquer pipeline, so it is slower than traditional NJ and requires the same input (plus constraint trees). The goal of NJMerge is to improve upon the accuracy of NJ by using a highly accurate but computationally intensive method to estimate subset trees. The effectiveness of this approach (in terms of accuracy) depends on both the model condition and the methods used to estimate subset trees, as discussed in Section 4.5.

In contrast, TreeMerge-fast does not require an $n \times n$ dissimilarity matrix and is faster than NJ, running in $O(n \log n)$ time instead of $O(n^3)$ time, within the divide-and-conquer pipeline. FastME, which builds a tree from a dissimilarity matrix in $O(n^2 \log n)$ time, could be used instead of NJ, but even so, the running time of the divide-and-conquer pipeline using TreeMerge-fast will be dominated by the time to estimate a starting tree as well as the time to perform preprocessing (discussed in the next section).

For users who are willing to forgo the guarantee of statistical consistency, the first step (of building a merge guide tree and subset decomposition) could be performed using less computationally intensive techniques (especially if these techniques are highly accurate in simulation studies). This opens up many avenues for algorithm design, and of course, determining whether these new techniques enable pipelines to be provably statistically consistent is another exciting avenue of future research. Other algorithmic developments may aid in this endeavor; for example, combining divide-and-conquer with iteration has been used successfully to improve robustness to the initial subset decomposition in a related application: co-estimating an MSA and ML (gene) tree [227, 233, 241, 242]. Iteration seems likely to be useful in the context of TreeMerge, as large collisions (i.e., collisions where TreeMerge-slow would run NJMerge on a large number of species) may indicate that starting over with a new subset decomposition and merge guide tree is necessary.

5.5.2 Computational Requirements of Species Tree Estimation Pipelines

Many biological studies (e.g., [24, 243]) have analyzed multi-locus datasets using ASTRAL and RAxML, so differences in their computational requirements are of interest. The timings we report for ASTRAL-III and RAxML are *not* directly comparable, because the ASTRAL-III timings do not include the time required for gene tree estimation. Using the same experimental set-up (a single Blue Waters compute node with 64 GB of memory and 16 floating-point cores), FastTree-2 took 3.6 hours on average to estimate 1 000 gene trees for datasets with 1 000 species (Supplementary Tables S4 and S5 in [220]), so in our study the amount time spent estimating 1 000 gene trees was small in comparison to the amount of time spent running ASTRAL-III on datasets with very high ILS (42 hours on average). However, the time required for gene tree estimation can vary greatly, depending on the method used and the analysis performed (e.g., analyses that search from multiple starting trees or perform non-parametric bootstrapping will be more computationally intensive).

It is possible that the bulk of the time could be spent estimating gene trees, but in some cases with very large numbers of species, the *entire* species tree estimation pipeline (which includes gene tree estimation) could be sped up by using TreeMerge-fast. Such a pipeline would proceed as follows.

1. Define pairwise disjoint subsets $S_1, S_2, \ldots, S_k$ of species and a merge guide tree $\mathcal{G}$.

2. Compute $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ and $\mathcal{D} = \{D^{i,j} : (i,j) \in \mathcal{G}\}$ as follows.

    - Root $\mathcal{G}$ at the node labeled 1 (i.e., the node corresponding to subset S_1).

- For every edge $e = (i, j)$ in a preorder traversal of G:
  - For each gene $g = \{1, 2, \ldots, m\}$:
    - Estimate an multiple sequence alignment (MSA) $A_g^{i,j}$ on species set $S_i \cup S_j$.
    - Estimate an maximum likelihood (ML) gene tree $P_g^{i,j}$ from $A_g^{i,j}$.
  - Compute the AGID distance matrix $D^{i,j}$ from $\mathcal{P}^{i,j} = \{\mathcal{P}_g^{i,j} : g \in \{1, \ldots, m\}\}$.
  - Estimate a constraint tree T_j from $\{P|_j : P \in \mathcal{P}^{i,j}\}$ using ASTRAL.
- Estimate constraint tree T_1 on $\mathcal{P}^{1,j}$ such that $(1, j)$ exists in $E(\mathcal{G})$.

3. Run TreeMerge on the input $(\mathcal{T}, \{\mathcal{D}, \mathcal{G}\})$.

This pipeline exploits locality at multiple levels: species trees do not need to be estimated on the full set of species, and similarly, gene trees and MSAs do not need to be estimated on the full set of species.

For users who want the guarantee of statistical consistency, the first step could be performed by using a statistically consistent method to build a starting tree T_0 on S and then using T_0 to define pairwise disjoint subsets of species and a merge guide as described in Theorem 5.5. The former could be achieved by running NJ on a dissimilarity matrix D of log-det distances (recall that log-det distances are statistically consistent under the MSC model). While this requires estimating an MSA for each gene on the full set S of species, it does not require gene tree estimation on the full set of species. This is critical as most (if not all) ML methods implement parallelism across sites in the MSA but *not* across species and therefore can be quite computationally intensive for very large numbers of species. In fact, a promising direction of future research is to explore divide-and-conquer pipelines using TreeMerge-fast in the context of gene tree estimation.

5.5.3 Future Studies

Future studies should investigate the robustness of TreeMerge-fast to the subset decomposition and the merge guide tree. Along these lines, other variations on the divide-and-conquer pipeline should be explicitly tested on a large collection of biological and simulated datasets. In particular, datasets should be simulated under challenging but biologically realistic model conditions, including those that produce model violations when estimating gene trees and/or species trees with popular methods. This is a condition where using TreeMerge may have an advantage, as some model assumptions may be preserved locally but not globally across a tree. In the context of gene tree estimation, changes in GC content can provide evidence that stationarity, an assumption of SRH models, is violated across the tree, but this model violation may be less significant locally. In addition, the substitution rate matrix can change across branches of the tree, producing a model violation referred to as *heterotachy*. There is a growing body of literature surrounding heterotachy [244, 245, 246, 247], and this, in turn, has sparked the development of new methods, such as GHOST [248]. These methods are more computationally intensive than traditional methods, so divide-and-conquer may be particularly beneficial in this context.

Of course, biologists not only want a tree topology, they also want estimates of uncertainty or estimates of the probability of error on each branch. This is commonly achieved through non-parametric bootstrapping, an approach that can easily be applied in the context of divide-and-conquer pipelines using TreeMerge-fast. However, as discussed in Section 3.4, model violations can result in highly supported false positive branches, so interpreting differences between species estimated on the same biological datasets can be challenging.

Since the time of our study, there have been many new developments in divide-and-conquer phylogeny estimation. Most notably, Le *et al.* [249] implemented a new algorithm for merging leaf-disjoint trees, called constrained-INC [250], and evaluated it within the context of gene tree estimation and species tree estimation, finding that it achieved comparable accuracy to NJMerge. Constrained-INC is more computationally efficient than NJMerge, so constrained-INC instead of NJMerge could be called during the local merge phase or during the global merge phase (to resolve collisions when using TreeMerge-slow). Lastly, divide-and-conquer pipelines using DTM methods could be compared to traditional divide-and-conquer pipelines (e.g., Disk Covering Methods) when robust implementations become publicly available for species tree estimation.

5.6 CONCLUSIONS

We presented TreeMerge, a new technique for merging leaf-label-disjoint trees that addresses two important limitations of NJMerge: first that NJMerge can fail and second that NJMerge is slow, running in $O(n^4k)$ time, where n is the number of species. In contrast, the serial version of TreeMerge-fast runs in $O(nk)$ time and the parallel version runs in $O(n \log k)$ time. This running time advantage came at the expense of restricting the solution space, but in our simulation study, there was little difference between the accuracy of TreeMerge-fast and NJMerge. Indeed, TreeMerge-fast was effective at scaling computationally intensive species tree estimation methods to large datasets. The impact was greatest for those datasets on which ASTRAL-III or RAxML failed to complete, either due to limited running time (for ASTRAL-III) or limited memory (for RAxML). When using genome-scale data, the computational requirements for ML analyses can be very large even for datasets with small numbers of species; for example, the Avian Phylogeomics Project with whole genomes for 48 birds used one TB of memory and took more than 200 CPU years to complete. Today, we hope that DTM methods, including TreeMerge-fast, make it computationally feasible for researchers with limited resources to analyze large multi-locus datasets—*phylogenomics for all*. In the future, we hope that the work here facilitates the development of highly parallel phylogeny estimation pipelines that have good empirical performance (accuracy) as well as good theoretical performance (statistical consistency).

This section contains the two plots presented in Section 5.4 Results.

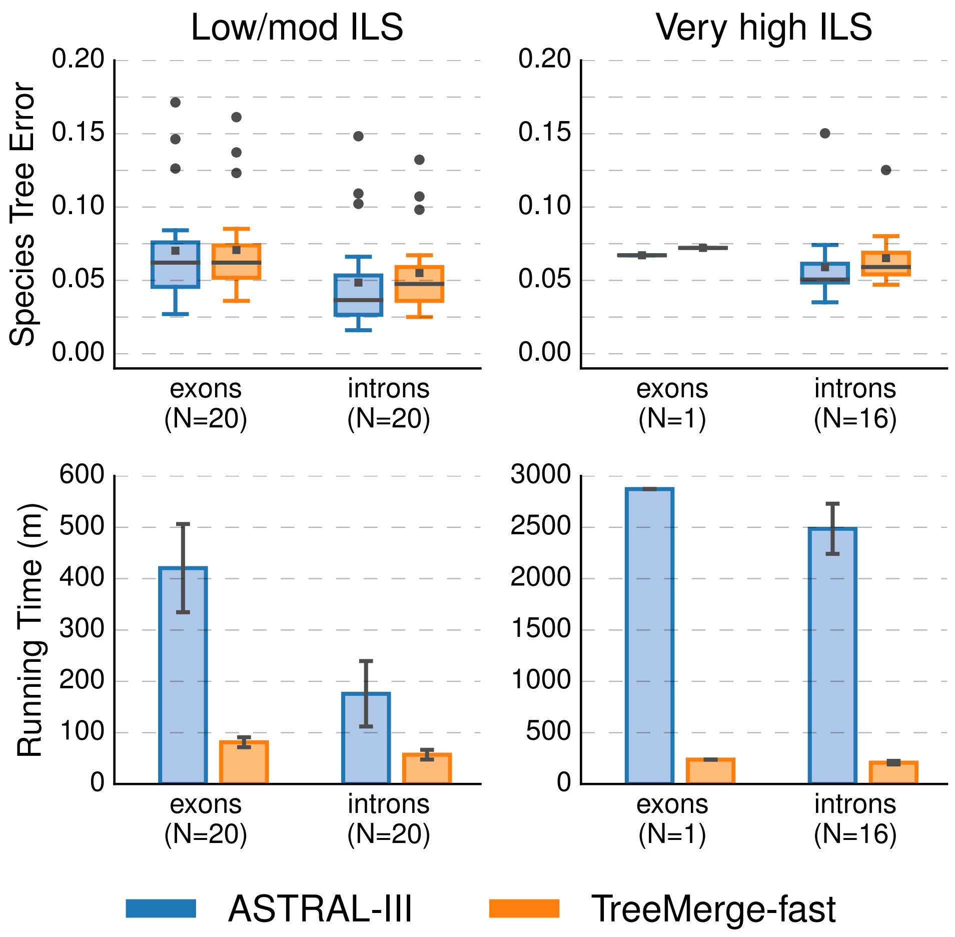


Figure 5.4: **Impact of using TreeMerge-fast with ASTRAL-III.** The top row shows species tree estimation error for datasets with 1 000 species and 1 000 genes. Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots extend from the first to the third quartiles, and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). The bottom row shows running time (in minutes); bars represent means, and error bars represent standard deviations across replicate datasets. The running time of TreeMerge-fast is the time to estimate the distance matrix, to estimate each subset tree using ASTRAL-III, and to combine the subset trees using TreeMerge-fast (Equation 5.1). The number N of replicates on which ASTRAL-III completed is shown on the x-axis; note that averages are taken across the replicates on which ASTRAL-III completed. When ASTRAL-III did not complete, it was due to running longer than the 48-hour maximum wall-clock time.

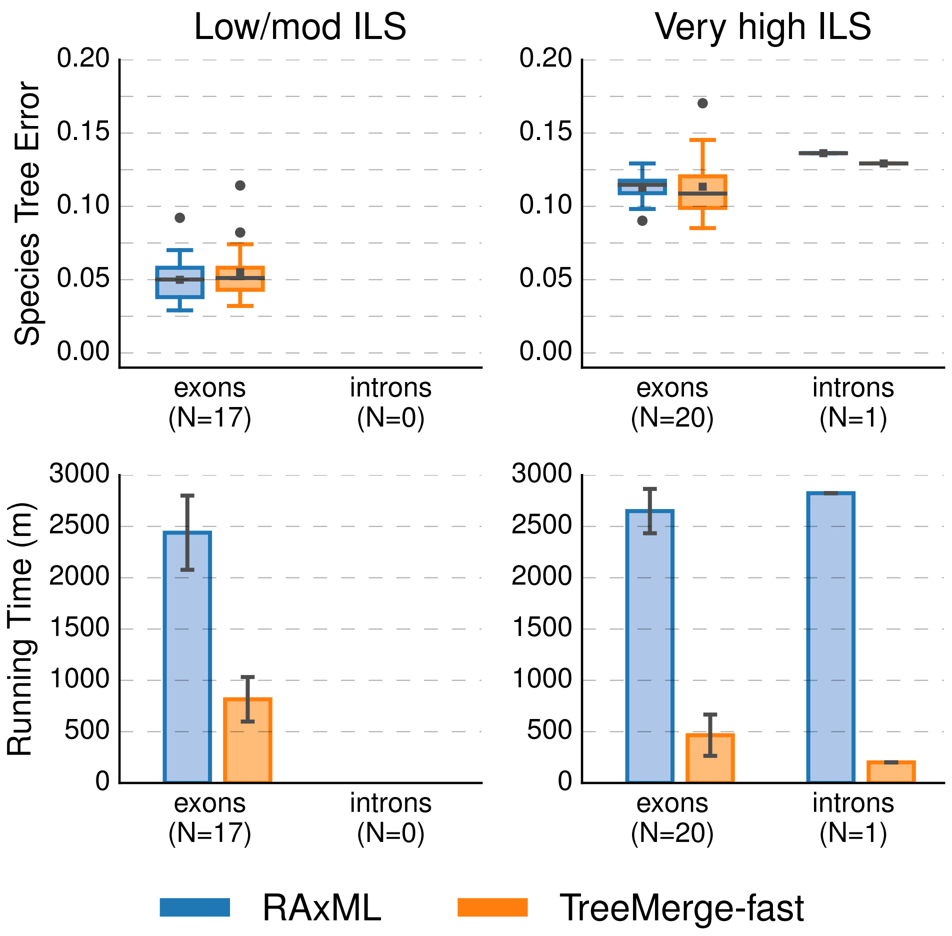


Figure 5.5: **Impact of using TreeMerge-fast with RAxML.** The top row shows species tree estimation error for datasets with 1 000 species and 1 000 genes. Gray bars represent medians, gray squares represent means, gray circles represent outliers, box plots extend from the first to the third quartiles, and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). The bottom row shows running time (in minutes); bars represent means, and error bars represent standard deviations across replicate datasets. The running time of TreeMerge-fast is the time to estimate the distance matrix, to estimate each subset tree using RAxML, and to combine the subset trees using TreeMerge-fast (Equation 5.1). The number N of replicates on which RAxML completed is shown on the x-axis; note that averages are taken across the replicates on which RAxML completed. When RAxML did not complete, it was due to Out Of Memory (OOM) errors; otherwise the last checkpoint written by RAxML was evaluated.

5.8 TABLES

This section contains the two tables presented in Section 5.4 Results.

Table 5.2: **Species tree error for NJMerge vs. NJMerge-2 vs. TreeMerge-fast.** A comparison of TreeMerge-fast to NJMerge and NJMerge-2 is shown below. Species tree estimation error (average ± standard deviation; maximum error is one) is shown for datasets with 1 000 species and 1 000 genes. The number of replicates on which NJMerge returned a tree is also shown, and averages are taken across the replicates on which NJMerge completed. When NJMerge failed to return a tree, it was due to algorithmic failure (i.e., considering a siblinghood proposal to be safe when it wasn't).

Level of ILS	Data Type	Number of Replicates	NJMerge	NJMerge-2	TreeMerge-fast
ASTRAL-III Analysis (i.e., M_T = ASTRAL-III, M_D = AGID)					
low/mod	exon	20	0.06 ± 0.03	0.06 ± 0.03	0.07 ± 0.03
low/mod	intron	20	0.05 ± 0.03	0.05 ± 0.03	0.06 ± 0.03
very high	exon	20	0.08 ± 0.04	0.08 ± 0.04	0.09 ± 0.03
very high	intron	20	0.06 ± 0.02	0.06 ± 0.02	0.06 ± 0.02
RAxML Analysis (i.e., M_T = RAxML, M_D = log-det)					
low/mod	exon	19	0.04 ± 0.02	0.05 ± 0.02	0.05 ± 0.02
low/mod	intron	20	0.03 ± 0.01	0.03 ± 0.01	0.04 ± 0.01
very high	exon	17	0.10 ± 0.01	0.10 ± 0.01	0.11 ± 0.01
very high	intron	18	0.09 ± 0.01	0.09 ± 0.01	0.10 ± 0.01

Table 5.3: **Running time for each step in divide-and-conquer pipelines using TreeMerge-fast.** The running time in minutes (mean ± standard deviation; maximum error is one) is given for each step in the divide-and-conquer pipeline broken down into the time to compute the distance matrix and the time to compute all subset trees and the time to merge the trees together using TreeMerge-fast (i.e., the three terms of Equation 5.1). We do not show the time required to compute the starting tree, which required just a few minutes for both RAxML and ASTRAL-III analyses.

Species Tree Height	Data Type	M_D	M_T on all subsets	TreeMerge $(D, \mathcal{T})$
ASTRAL-III Analysis (i.e., M_T = ASTRAL-III, M_D = AGID)				
low/mod	exon	1 ± 0	49 ± 7	32 ± 5
low/mod	intron	1 ± 0	26 ± 7	31 ± 5
very high	exon	1 ± 0	216 ± 23	33 ± 6
very high	intron	1 ± 0	178 ± 19	34 ± 4
RAxML Analysis (i.e., M_T = RAxML, M_D = log-det)				
low/mod	exon	24 ± 1	801 ± 228	34 ± 4
low/mod	intron	34 ± 3	1333 ± 301	32 ± 4
very high	exon	23 ± 2	412 ± 199	30 ± 5
very high	intron	40 ± 3	752 ± 442	30 ± 4

CHAPTER 6: SPECIES TREE ESTIMATION IN THE PRESENCE OF GENE DUPLICATION AND LOSS WITH FASTMULRFS

This chapter contains material perviously published in "FastMulRFS: Fast and accurate species tree estimation under generic gene duplication or loss models" [251], which was joint work with T. Warnow. All supplementary materials referenced in this chapter are freely available on Dryad: doi.org/10.1101/835553. Datasets and software commands necessary to reproduce this study are freely available on the Illinois Data Bank: doi.org/10.13012/B2IDB-5721322_V1. FastMulRFS *is freely available on Github: github.com/ekmolloy/fastmulrfs. Note that plots and tables appear at the end of this chapter in Sections 6.7 and 6.8, respectively.*

6.1 INTRODUCTION

In the previous chapters, we discussed species tree estimation in the presence of incomplete lineage sorting (ILS), where gene tree heterogeneity is the result of genealogical relationships between alleles. Gene duplication and loss (GDL) is another major source of gene tree heterogeneity (one that is expected to be common in fungi [32] and plants [25]). However, most species tree estimation methods, including the ones discussed thus far in this dissertation (Section 2.7), are designed for orthologous genes. Because orthology detection is still difficult to do correctly [35, 36, 37] and mistakes in orthology prediction can result in incorrect species trees, multi-copy genes are often excluded from species tree estimation (e.g., [24, 25]). Methods that can estimate species trees from gene families are of increasing interest, as this would not only avoid the challenges of orthology detection but also enable the phylogenetic signal in multi-copy genes to be leveraged during species tree estimation.

Several methods have been proposed to estimate species trees from multi-copy genes. The most well-known method explicitly based on a parametric model of GDL is probably PHYLDOG [212], which uses likelihood to co-estimate the species tree and gene family trees. PHYLDOG is very computationally intensive and so is limited to very small datasets with 10 or so species. Recently, De Oliveira Martins *et al.* [252] proposed a Bayesian supertree method *guenomu* for multi-copy gene trees. Because *guenomu* requires a posterior distribution to be estimated for each gene family tree (e.g., using MrBayes [203]) as input, it is not fast enough to use on genome-scale datasets with 100 or more species.

Non-parametric methods are more commonly used alternatives. For example, *Gene Tree Parsimony (GTP)* methods take a set of (estimated) gene family trees as input, and then seek a species tree that implies the minimum number of evolutionary events, such as gene duplications and gene losses. Examples of GTP methods include DupTree [61], iGTP [253], and DynaDup [254]. Since GTP is NP-hard, most of these methods operate by using hill-climbing. DynaDup, in contrast, uses dynamic programming (DP) to find an optimal solution within a constrained search space; this type of approach, to the best of our knowledge, was first proposed in [48] and has since been utilized for other problems (note that this is the type of approach used by ASTRAL to solve the bipartition-constrained maximum quartet support supertree (MQSS) problem [49, 78]). Although GTP methods can be computationally intensive, they are more scalable than other approaches, and several phylogenomic studies have used GTP methods to analyze biological datasets (e.g., [255, 256]).

Other fast approaches include supertree methods that have been adapted to work with gene family trees, referred to as MUL-trees. The most well known supertree method for MUL-trees is perhaps MulRF [47], which attempts to find a solution to the NP-hard Robinson-Foulds Supertree problem for MUL-trees (RFS-MUL-trees). Although MulRF does not explicitly account for GDL, it has been shown to produce more accurate species trees than DupTree and iGTP on datasets simulated under challenging model conditions with gene tree heterogeneity due to GDL, ILS, horizontal gene transfer (HGT), and gene tree estimation error (GTEE) [257].

In a very recent advance, Legried *et al.* [57] proved that ASTRAL-multi [58], an extension of ASTRAL [49] to address multi-allele inputs, is statistically consistent under the probabilistic model of GDL proposed by Arvestad *et al.* [258] (recall that gene trees evolve i.i.d. within a species tree with a duplication rate and a loss rate fixed across the edges of the species tree; see Section 2.5.2). In fact, ASTRAL-multi is the only method that has been proven statistically consistent under any GDL model. Yet, the experimental study comparing ASTRAL-multi to three earlier species tree estimation methods, including DupTree, STAG [259], and MulRF, showed that ASTRAL-multi had good but not exceptional accuracy. When the duplication and loss rates were both high, ASTRAL-multi was less accurate than MulRF (although ASTRAL-multi was more accurate than STAG and typically more accurate than DupTree except when GTEE was low). The high accuracy of MulRF in comparison to ASTRAL-multi encouraged us to explore the optimization problem that MulRF attempts to solve.

In the remainder of this chapter, we prove that the true species tree is an optimal solution to the RFS-MUL-trees problem, provided there is no adversarial GDL (which occurs when the pattern of duplication events and loss events produces bipartitions that are incompatible with the species tree). This model is less restrictive than the probablistic GDL model in that it does not assume genes evolve *i.i.d.* (similar to the No Common Mechanism (NCM) model) but is more restrictive in that it prohibits adversarial GDL. However, we conjecture that adversarial GDL will occur with sufficiently low probability so that an exact solution to the RFS-MUL-trees problem will be statistically consistent for reasonable duplication and loss probabilities. This result is enabled by proving that, when solving the RFS-MUL-trees problem, any input set of MUL-trees can be replaced by a set of singly-labeled trees. By combining this reduction technique with DP within a constrained search space, we develop a new method, *FastMulRFS*. As we will show, FastMulRFS is statistically consistent under a generic GDL model that prohibits adversarial GDL.

Lastly, we compare FastMulRFS to ASTRAL-multi, DupTree, and MulRF on datasets simulated under the DLCoal model with varying levels of GDL, ILS, and GTEE. As we will show, FastMulRFS is generally more accurate than DupTree and ASTRAL-multi and ties for most accurate with MulRF. In addition, FastMulRFS is much faster than MulRF and ASTRAL-multi and ties for fastest with DupTree. The improvement in performance over ASTRAL-multi is the most important result, as ASTRAL-multi is the only other method to date that has been proven statistically consistent under a probabilistic GDL model. In summary, FastMulRFS is a fast method for species tree estimation that does not require reliable orthology detection and outperforms the leading alternative methods (even under conditions for which FastMulRFS is not yet established to be statistically consistent).

6.2 APPROACH

We begin by extending some of the terminology and definitions from Sections 2.1—2.3 to MUL-trees. Recall that a *phylogenetic tree* T is defined by the triplet (t, S, ϕ), where t is a tree, S is a set of labels, and $\phi : L(t) \rightarrow S$ assigns labels to the leaves of t. If ϕ is a bijection, we say that T is singly-labeled; otherwise, we say that ϕ is multi-labeled or equivalently that T is a MUL-tree. Recall that deleting an edge e but not its endpoints from T produces two subtrees t_A and t_B, defining two label sets: $A = \{\phi(l) : l \in L(t_A)\}$ and $B = \{\phi(l) : l \in L(t_B)\}$. When T is singly-labeled, every edge $e \in E(T)$ splits the leaf labels into two sets A and B such that $A \cap B = \emptyset$; this does not hold when there are two leaves in T with the same label. Lastly, recall that the RF distance (i.e., the edit distance under contraction and refinement operations) between two singly-labeled trees on the same label set can be computed as the bipartition distance (Theorem 2.1). Theorem 2.1 does not hold when one or both trees is a MUL-tree.

6.2.1 Robinson-Foulds Supertree problem for MUL-trees

To present the RF supertree problem for MUL-trees, we need three additional definitions from Ganapathy *et al.* [260] and Chaudhary *et al.* [46].

Definition 6.1 (Full Differentiation)**.** We say that $M' = (m, S', \phi')$ is a *full differentiation* of MUL-tree $M = (m, \phi, S)$ if $\phi' : L(m) \to S'$ is a bijection. In other words, M' is a singly-labeled version of M.

Definition 6.2 (Mutually Consistent Full Differentiations)**.** Let $M_1' = (m_1, S', \phi_1')$ and $M_2' = (m_2, S', \phi_2')$ be full differentiations of MUL-trees $M_1 = (m_1, S, \phi_1)$ and $M_2 = (m_2, S, \phi_2)$, respectively. For $i = 1, 2$, we define $R_i(s) \subseteq S'$ to be the set of labels given to the leaves in M_i' that are labeled s in M_i. We say that M_1' and M_2' are *mutually consistent full differentiations (MCFDs)* of M_1 and M_2 if $R_1(s) = R_2(s)\ \forall s \in S$.

Ganapathy *et al.* [260] showed that if M_1 and M_2 are both MUL-trees, then their RF distance can be computed as

$$MulRF(M_1, M_2) := \min\{RF(M_1', M_2') : M_1', M_2' \text{ are MCFDs of } M_1, M_2\} \tag{6.1}$$

which implies an exponential-time algorithm for computing the RF distance between two MUL-trees [260], Later, this problem was proven to be NP-complete by Chaudhary *et al.* [46].

Chaudhary *et al.* [46] also introduced a special case, where one of the two MUL-trees has the following property: every leaf with the same label is grouped together in a polytomy that is separated by an edge from the rest of the tree. A MUL-tree with this property can be viewed as an extended version of a singly-labeled tree.

Definition 6.3 (Extended Version)**.** Let $T = (t, S, \phi_T)$ be a singly-labeled tree, let $M = (m, S, \phi_M)$ be a MUL-tree, and let $k(s)$ be the number of leaves with label s in M. The *extended version* of T with respect to M, denoted $Ext(T, M)$, is created by attaching $k(s)$ new leaves to the leaf labeled s in T, assigning label s to each of these new leaves, and then repeating this process for all $s \in S$.

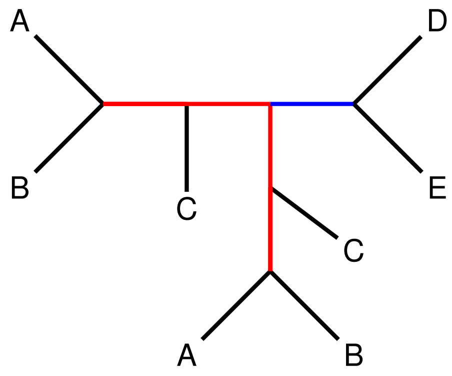
Chaudhary *et al.* [46] showed that the RF distance between a MUL-tree M and an extended version of a singly-labeled tree $Ext(T, M)$ can be computed in polynomial time. Based on this result, they proposed the Robinson-Foulds Supertree problem for MUL-trees (RFS-MUL-trees).

Definition 6.4 (Robonsin-Foulds Supertree Problem for MUL-trees)**.** Let $\mathcal{T}$ be a set of MUL- trees. If a tree T^* on label set $S = \bigcup_{M \in \mathcal{T}} S(M)$ is in the set

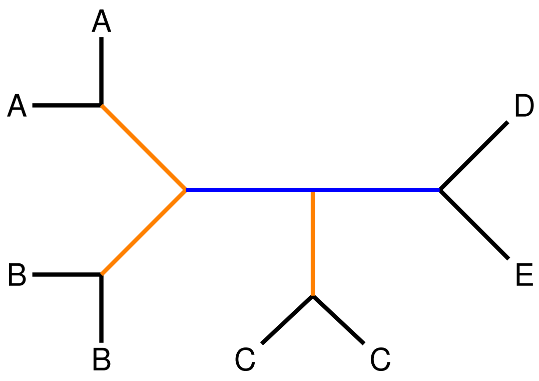
$$\sum_{M \in \mathcal{P}} RF(Ext(T|_{S(M)}, M)', M') \tag{6.2}$$

then we say that T^* is an *RFS-MUL-trees* supertree for $\mathcal{T}$.

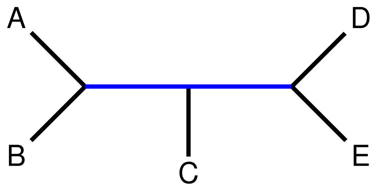
Note that when $\mathcal{P}$ is a profile of singly-labeled trees, then the RFS-MUL-trees problem is equivalent to the Robinson-Foulds supertree (RFS) problem.



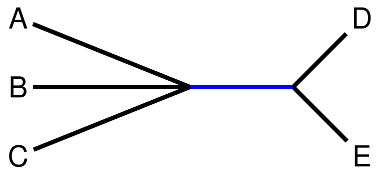
(b) Mul-tree M



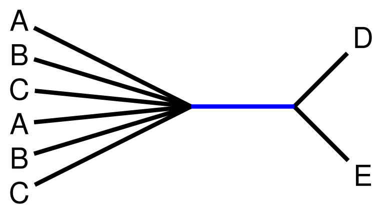
(c) $Ext(T, M)$



(a) Singly-labeled tree T



(e) $\mathcal{R}(\mathcal{X}(M))$



(d) $\mathcal{X}(M)$

Figure 6.1: **Reduction of the RFS-MUL-trees problem to the RFS problem.** Subfigure (a) shows a candidate RFS-MUL-trees supertree T for a set $\mathcal{P}$ of MUL-trees, and subfigure (b) shows a MUL-tree M in P; note that both T and M are on the same label set $S = \{A, B, C, D, E\}$. To compute the RF distance between T and M, we build the extended version of T with respect to M (Definition 6.3), producing $Ext(T, M)$, as shown in subfigure (c). The trivial edges in $Ext(T, M)$ (shown in orange) exist in *any possible* singly-labeled, binary tree on S, so these edges do not impact the solution to the RFS-MUL-trees problem. Similarly, the edges in MUL-tree M that are shown in red cannot exist in an extended version of *any possible* singly-labeled, binary tree on S, so these edges do not impact the solution to the RFS-MUL-trees problem. We contract all internal edges in M with the property that there are at least two leaves, one on either side of the edge, with the same label; this produces $\mathcal{X}(M)$, as shown in subfigure (d). Furthermore, because all leaves with the same label are now on the same side of *every* edge in $\mathcal{X}(M)$, we can delete all but one leaf with each label; this produces $\mathcal{R}(\mathcal{X}(M))$, as shown in subfigure (e). The resulting tree is a non-binary, singly-labeled tree on S, so we can compute the RF distance between T and $\mathcal{R}(\mathcal{X}(M))$ using Equation 2.1. These observations are formalized in Lemma 6.1.

6.2.2 Reducing from MUL-trees to singly-labeled trees

We simplify the RFS-MUL-trees problem by providing an alternative proof that the RF distance between between a singly-labeled tree T and a MUL-tree M can be computed (up to a constant factor that does not depend on T) in polynomial time by reducing the MUL-trees to a set of singly-labeled trees; see Figure 6.1 for intuition behind this proof.

To begin, we define two transformations that can be applied to a MUL-tree $M = (m, S, \phi)$ or to its full differentiation $M' = (m, S', \phi')$ by using the function $f : S' \to S$ with the property that $f(\phi'(l)) = \phi(l)$ for all $l \in L(m)$.

Definition 6.5 (Contracted Version)**.** The *contracted version* of M, denoted $\mathcal{X}(M)$, is created by contracting every internal edge e with the property that there are at least two leaves, one on either side of the edge, with the same label. Similarly, the contracted version of M', denoted $\mathcal{X}(M')$, is created by contracting every internal edge e with $\pi(e) = A|B$ such that $f(A) \cap f(B) \neq \emptyset$.

Definition 6.6 (Reduced Version)**.** If all leaves with species label s are on the same side of *every* internal edge in $E(M)$, then they can be represented by a single leaf labeled s. The *reduced version* of M or M', denoted $\mathcal{R}(M)$ or $\mathcal{R}(M')$, respectively, is created as follows. For every $s \in S$ with the aforementioned property, delete all but one of the leaves in the set $\{f(\phi'(l)) = \phi(l) = s : l \in L(m)\}$ (suppressing internal nodes of degree 2) and relabel the remaining leaf s.

It is easy to see that $\mathcal{R}(\mathcal{X}(M'))$ is a singly-labeled tree that is isomorphic to $\mathcal{R}(\mathcal{X}(M))$, because after applying the function Σ to either M' or M, all the leaves with species label s will be on the same side of every edge and thus can be replaced by a single leaf with species label s by applying the function $\mathcal{R}$. This observation holds for all $s \in S$.

Lemma 6.1. *Let T be an unrooted, singly-labeled, and fully resolved tree on label set S, let $M = (m, S, \phi)$ be an unrooted MUL-tree, and let $Ext(T, M)'$ and $M' = (m, S', \phi')$ be MCFDs of $Ext(T, M)$ and M, respectively. Then,*

$$RF(Ext(T, M)', M') = RF(T, M_X) + K \tag{6.3}$$

where $M_X = \mathcal{R}(\Sigma(M))$ and K is a constant that does not depend on the topology of the singly-labeled tree T on S.

Proof. Let $f : S' \to S$ be a function with the property that $f(\phi'(l)) = \phi(l)$ for all $l \in L(m)$.

Now we define the following bipartition sets.

$$X = \{A|B \in Bip(M') : f(A) \cap f(B) \neq \emptyset\} \tag{6.4}$$
$$R = \{A|B \in Bip(M') \setminus X : |A| > 1, |B| > 1, \text{ and either } |f(A)| = 1 \text{ or } |f(B)| = 1\} \tag{6.5}$$

It is easy to see that X contains bipartitions that *cannot exist* in $Bip(Ext(T, M)')$ for any singly-labeled tree T on S and that R contains bipartitions that *must exist* in $Bip(Ext(T, M)')$ for any singly-labeled tree T on S. (note that edges that induce bipartitions in the set X are colored red in Figure 6.1b and edges that induce bipartitions in the set R are colored orange in Figure 6.1c.) Let E' denote $Ext(T, M)'$. Then,

$$\begin{aligned} |Bip(E') \cap Bip(M')| &= |Bip(E') \cap Bip(\mathcal{X}(M'))| \\ &= |Bip(\mathcal{R}(E')) \cap Bip(\mathcal{R}(\mathcal{X}(M')))| + |R| + |L(m)| - |S| \\ &= |Bip(T) \cap Bip(M_X)| + |R| + |L(m)| - |S| \\ &= 0.5\big[|E(M_X)| + |E(T)| - RF(T, M_X)\big] + |R| + |L(m)| - |S| \\ &= 0.5\big[|E(M_X)| + 2|S| - 3 - RF(T, M_X)\big] + |R| + |L(m)| - |S| \\ &= 0.5\big[|E(M_X)| - 3 - RF(T, M_X)\big] + |R| + |L(m)| \end{aligned} \tag{6.6}$$

Let c be the number of species in $S(M)$ that have multiple copies (i.e., $c = |\{s \in S(M)\}|$). Then,

$$\begin{aligned} RF(E', M') &= |E(E')| + |E(M')| - 2|Bip(E') \cap Bip(M')| \\ &= \big(|S| - 3 + c + |L(m)|\big) + |E(m)| - 2|Bip(E') \cap Bip(M')| \\ &= RF(T, M_X) + |S| + c + |E(m)| - |E(M_X)| - 2|R| - |L(m)| \end{aligned} \tag{6.7}$$

where S, c, $E(m)$, $E(M_X)$, R, and $L(m)$ are independent of T. QED.

In Lemma 6.1, we show that $MulRF(Ext(T, M), M)$ can be computed in polynomial time, because computing $RF(Ext(T, M)', M')$ does not depend on the MCFDs of $Ext(T, M)$ and M. In addition, we show that the RF distance between $Ext(T, M)$ and M can be computed (up to a constant factor that does not depend on the topology of a singly-labeled tree T on S) by simply transforming M into a (potentially unresolved) singly-labeled tree on S and computing its RF distance from T.

The following theorem easily follows from Lemma 6.1.

Theorem 6.1. Let $\mathcal{P}$ be a set of unrooted MUL-trees, let $\mathcal{P}_X = \{\mathcal{R}(\mathcal{X}(M)) : M \in \mathcal{P}\}$, and let T be an unrooted, singly-labeled, and fully resolved tree on label set $S = \bigcup_{M \in \mathcal{P}} S(M)$.

Then, T is an RFS-MUL-trees supertree for $\mathcal{P}$ if and only if T is an RF supertree for $\mathcal{P}_X$.

Definition 6.7 (Valid and Invalid Bipartitions)**.** Let $M = (m, S, \phi)$ be an unrooted MUL-tree. Suppose that deleting an edge e but not its endpoints from M produces two subtrees $m_A(e)$ and $m_B(e)$, defining the label sets: $A = \{\phi(l) : l \in L(M_A)\}$ and $B = \{\phi(l) : l \in L(M_B)\}$. If $A \cap B = \emptyset$, we say that e induces a *valid bipartition* and allow $\pi(e) = A|B$ to be contributed to the set $Bip(M)$. If $A \cap B \neq \emptyset$, we say that e induces an *invalid bipartition* and do *not* allow $\pi(e) = A|B$ to be contributed to the set $Bip(M)$; alternatively, we say that e fails to induce a bipartition.

Theorem 6.2. Let $\mathcal{P}$ be a set of unrooted MUL-trees, and let T be an unrooted, singly-labeled, and fully resolved tree on label set $S = \bigcup_{M \in \mathcal{P}} S(M)$. By combining Lemma 6.1 and Definition 6.7, it is easy to see that T is in the set:

$$\sum_{M \in \mathcal{P}} RF(T|_{S(M)}, M) \tag{6.8}$$

if and only if T is an RFS-MUL-trees supertree for $\mathcal{P}$.

6.2.3 FastMulRFS

A consequence of Theorem 6.1 is that any heuristic for the RFS problem can be used for the RFS-MUL-trees problem simply by computing $\mathcal{P}_X$ prior to running the heuristic. In this study, we explore the impact of using FastRFS [59], an effective heuristic for the RFS problem, which solves the bipartition-constrained version of the RFS problem exactly using DP. We refer to this pipeline as FastMulRFS.

- **FastMulRFS Input**: Set $\mathcal{P} = \{M_1, M_2, \ldots, M_k\}$ of unrooted MUL-trees
- **FastMulRFS Output**: An unrooted, fully resolved phylogenetic tree T on label set $S = \bigcup_{i=1}^{k} S(M_i)$ such that $\sum_{i=1}^{k} RF(T|_{S(M_i)}, M_i)$ is maximized and $Bip(T) \subseteq \Sigma$ (note that Σ is a set of allowed bipartitions computed from $\mathcal{P}$)

The set Σ is the space that FastRFS required to run FastRFS, and the current implementation of FastRFS runs ASTRAL-III to compute Σ. This leads to the following pipeline.

In summary, FastMulRFS runs in $O(mnk + nk|\Sigma|^2)$ time, where n is the number of species, k is the number of MUL-trees, and m is the largest number of leaves in any of the MUL-trees. The default technique for constructing the set Σ of allowed bipartitions enforces

Algorithm 6.1: FastMulRFS.

```
Input  : Set $\mathcal{P} = \{M_1, M_2, \ldots, M_k\}$ of unrooted MUL-trees
Output: An unrooted, fully resolved phylogenetic tree $T$ on $S = \bigcup_{i=1}^{k} S(M_i)$ s.t.
        $\sum_{i=1}^{k} RF(T|_{S(M_i)}, M_i)$ is maximized and $Bip(T) \subseteq \Sigma$

Function FastMulRFS($\mathcal{P}$):

    Step 1: Use Algorithm 6.2 to create $\mathcal{P}_X = \{\mathcal{R}(\mathcal{X}(M)) : M \in \mathcal{P}\}$ in $O(mnk)$ time,
    where $n = |S|$ and $m$ is the largest number of leaves in any MUL-tree in $\mathcal{P}$.
    $\mathcal{P}_X \leftarrow$ ContractThenReduceMultree($\mathcal{P}$)

    Step 2: Use ASTRAL-III [261] to produce a set $\Sigma$ of allowed bipartitions s.t.
    $|\Sigma| = O(nk)$. By default, $\Sigma$ includes the bipartition set induced by every tree in
    $M_X \in \mathcal{P}_X$ such that $S(M_X) = S$. Other bipartitions may be added to $\Sigma$ to guarantee
    at least one fully resolved tree $T$ satisfies $Bip(T) \subseteq \Sigma$ and to improve accuracy by
    expanding the space of allowed solutions. The running time of ASTRAL-III is
    $O(nk|\Sigma|^{1.726})$; we run ASTRAL-III to construct $\Sigma$ and then exit.
    $\Sigma \leftarrow$ASTRAL($\mathcal{P}_X$)

    Step 3: Run FastRFS [59] in $O(nk|\Sigma|^2)$ time.
    $T \leftarrow$ FastRFS($\mathcal{P}_\mathcal{X}$, $\Sigma$)

    return $T$
```

$|\Sigma| = O(nk)$ and as we will show, this suffices for proofs of statistical consistency under generic GDL models when no adversarial GDL occurs.

6.2.4 Species Tree Estimation using FastMulRFS

Generic GDL models: Our generic GDL models proceed in the same fashion as the probabilistic GDL models proposed by Arvestad *et al.* [258]; however, instead of having one duplication rate λ and one loss rate μ that is fixed across every branch of the species tree and across every gene, we allow each gene g and each edge e to have its own duplication rate $\lambda(e, g)$ and loss rate $\mu(e, g)$; in this way, our generic GDL model is similar to the NCM model. It is easy to see that our generic GDL models contain the probabilistic GDL models of [258] as sub-models. Recall that under these GDL models, duplications and losses follow a Poisson process. Let $N(e, g)$ denote the number of events (either duplications or losses) on edge e for gene g, and let $t(e)$ denote the length of the edge e in time units. Then, for gene g, the probability of n events on edge e is

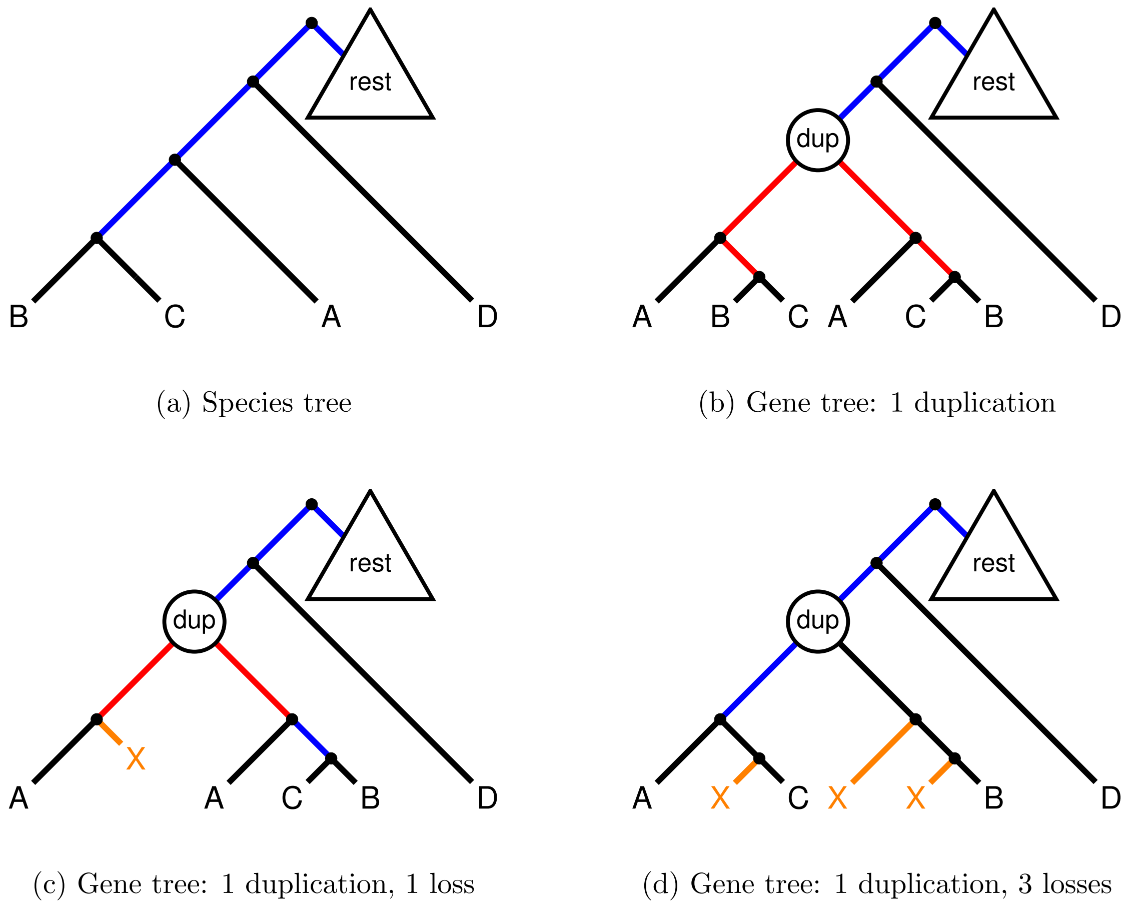
$$P\Big(N(e,g)=n\Big)=\frac{1}{n!}\bigg(\big(\lambda(e,g)+\mu(e,g)\big)\cdot t(e)\bigg)^{n}\times exp\bigg(-\big(\lambda(e,g)+\mu(e,g)\big)\cdot t(e)\bigg) \quad (6.9)$$

Clearly, the probability of no duplication/loss events (i.e., $n = 0$) is strictly greater than zero for every edge e and every gene g.

Adversarial GDL: We say that *adversarial GDL* has occurred when the gene evolution process produces a gene family tree with a bipartition π that is *not* compatible with the true species tree T^* (Definition 2.4). Adversarial GDL requires a sequence of events (a duplication followed by a carefully selected set of losses) that coordinate to produce such a bipartition. Figure 6.2d illustrates a scenario where adversarial GDL occurs: a gene duplicates on the edge above Y, the most recent common ancestor (MRCA) of species $\{A, B, C\}$, in the species tree (Figure 6.2a), so that Y has two copies of the gene. Then, one copy of the gene is lost on the edge above B, whereas the other copy of the gene is lost on the edge above A and on the edge above C. As a result, the gene family tree shown in Figure 6.2d is singly-labeled, but the gene family tree induces a bipartition $(A, C|B, D)$ that is incompatible with the species tree; by definition, this is adversarial GDL.

Figure 6.2b illustrates an alternative scenario where the same duplication event is followed by one copy of the gene being lost on the branch above the MRCA of species $\{B, C\}$. In this case, not only is there no adversarial GDL, but also the gene family tree induces a bipartition $(A, D|B, C)$ that is compatible with the species tree. Because one of the species retains both copies of a gene, the two arcs that are incident to the duplication node fail to induce bipartitions, as they are on the path between two leaves with the same label. Furthermore, suppose that A, B, and C are clades (rather than leaves), then every edge in the two A clades (and the edges on the path connecting the two A clades) would fail to induce a bipartition (assuming no other loss events). In contrast, every edge in the B clade and the C clade would induce a bipartition that is compatible with the species tree (assuming no other duplication events).

In some sense, duplication events hide bipartitions, while losses (following a duplication event) can reveal bipartitions. A carefully selected pattern of losses (after the duplication) can result in adversarial GDL (i.e., a particular bipartition π that is not in the species tree), but small changes to that pattern may well produce bipartitions that are in the true species tree or are incompatible with π. Therefore, while adversarial GDL may occur, it may not have high impact on tree estimation based on the RFS-MUL-trees criterion.



(a) Species tree

(b) Gene tree: 1 duplication

(c) Gene tree: 1 duplication, 1 loss

(d) Gene tree: 1 duplication, 3 losses

Figure 6.2: **Impact of duplications and losses on RFS-MUL-trees.** Subfigure (a) shows a rooted species tree and subfigures (b)–(d) show three rooted gene family trees that evolved within the species tree. Internal edges that induce valid bipartitions are shown in blue, internal edges that induce invalid bipartitions are shown in red, terminal edges are shown in black, and losses are shown in orange. Subfigure (b) shows a gene family tree with a duplication event on the edge above the MRCA of species $\{A, B, C\}$, which we refer to as species Y. All internal edges below the duplication fail to induce bipartitions; therefore, they do not impact the solution space for RFS-MUL-trees. Subfigure (c) shows a gene family tree with the same duplication event followed by one copy of the gene being lost from the MRCA of species $\{B, C\}$. Because A retains both copies, the internal edges on the path between A (on the left) and A (on the right) fail to induce bipartitions; therefore, they do not impact the solution space for RFS-MUL-trees. As long as one species retains both copies of a gene, the two arcs that are incident to the duplication node fail to induce bipartitions, as they must be on the path between two leaves with the same label. Subfigure (d) shows a gene family tree with the same duplication event followed by one copy of the gene being lost from species B and the other copy of the gene being lost from both species A and C. None of the species that evolved from Y retain both copies of the gene, so all edges below the duplication node induce valid bipartitions. Because this scenario produces a valid bipartition that is incompatible with the species tree, we refer to this situation as *adversarial GDL*.

In the remainder of this section, we will discuss model conditions under which adversarial GDL cannot occur: the *duplication-only* case, where all genes evolve with duplication but no loss, and the *loss-only* case, where all genes evolve with loss but no duplication. To prove that a model condition prohibits adversarial GDL, we need to establish that any bipartition that appears in a gene family tree is compatible with the species tree. Equivalently, any complete bipartition (i.e., a bipartition on the full species set) induced by any gene family tree must also be induced by the species tree, and any incomplete bipartition (i.e., a bipartition on a proper subset of the species set) induced by any gene family tree must be able to be extended (by adding the missing species) so that it becomes a bipartition induced by the species tree (Definition 2.4). It is trivial to see that if a gene evolves only with losses, then there is no adversarial GDL for that gene (Lemma 6.2), but the proof for duplication-only evolution is more interesting (Lemma 6.3).

Lemma 6.2. Let $\mathcal{P}$ be a set of true gene trees that evolved within the rooted species tree T^* under a stochastic loss-only model of gene evolution. Then, for $\pi \in \{Bip(M) : M \in \mathcal{P}\}$, π is compatible with T^*. Hence, loss-only models have no adversarial GDL.

Lemma 6.3. Let $\mathcal{P}$ be the set of true gene trees that evolved within the rooted species tree T^* under a stochastic duplication-only model of gene evolution. Then for every MUL-tree $M \in \mathcal{P}$, $Bip(M) \subseteq Bip(T^*)$. Hence, duplication-only models have no adversarial GDL.

Proof. Let M be an unrooted version of a MUL-tree in $\mathcal{P}$, and let e be an internal edge in $E(M)$. We say that an internal edge $e = (x, y) \in E(M)$ lies below a duplication node if there is at least one duplication node on the path from either x or y to the root in the rooted version of M; otherwise, we say that e is above all duplication nodes. An internal edge e is below a duplication node if and only if there is at least one species on both sides of e. It follows that any internal edge e below a duplication node will be contracted when producing $\mathcal{X}(M)$. Conversely, an internal edge e is above all duplication nodes if and only if there are no leaves on both sides of e with the same species label. It follows that any internal edge e above all duplication nodes will *not* be contracted when producing $\mathcal{X}(M)$. Finally, consider a bipartition induced by an edge that is not contracted, and therefore has no duplication nodes above it. This bipartition appears in the true species tree T^*, since the only events that cause the gene family tree to differ from the true species tree are duplications. QED.

We now prove that FastMulRFS is statistically consistent under generic GDL models if no adversarial GDL occurs.

Theorem 6.3. The true species tree T^* is an RFS-MUL-trees supertree for any input $\mathcal{P}$ for which no adversarial gene duplication and loss occurred.

Proof. The optimization problem seeks a binary tree T that minimizes the sum of the RF distances to the input MUL-trees; this is equivalent to maximizing the sum of the number of compatible bipartitions in the input MUL-trees. If no adversarial GDL occurs, then by definition, every bipartition in the input MUL-trees is compatible with the true species tree T^*, and so T^* is an optimal solution to the RFS-MUL-trees problem. QED.

Theorem 6.4. FastMulRFS is statistically consistent under a generic GDL model for which adversarial GDL is prohibited.

Proof. Let T^* be the true species tree. By Theorem 6.3, T^* is an optimal solution to the RFS-MUL-trees problem for any input $\mathcal{P}$ for which no adversarial GDL occurred, so as the number of genes increases, T^* is an optimal solution to the RFS-MUL-trees problem. FastMulRFS finds an optimal solution to the RFS-MUL-trees problem subject to the output tree T satisfying $Bip(T) \subseteq \Sigma$ (Theorem 3 in [59] and Theorem 6.1). Our generic GDL models assume that the probability of no duplication or loss occurring on an edge is always greater than zero for every gene, so there is a strictly positive probability of the true species tree T^* appearing in the set $\mathcal{P}$ of gene family trees; therefore, as the number of genes increases, Σ (as constructed by the default setting within FastMulRFS) will contain all bipartitions in the set $Bip(T^*)$ with probability converging to one. It follows that, as the number of genes goes to infinity, the probability that FastMulRFS will return T^* converges to one. QED.

We conclude this section with a conjecture.

Conjecture 6.1. FastMulRFS is statistically consistent under a generic model of GDL for probabilities of gene duplication and loss, so that adversarial GDL has sufficiently low probability.

6.3 PERFORMANCE STUDY

We evaluated FastMulRFS in comparison to ASTRAL-multi, DupTree, and MulRF on biological and simulated datasets, considering species tree topological accuracy and running time.

6.3.1 Fungal Dataset

We analyzed a fungal dataset with 16 species and 5 351 genes from Rasmussen and Kellis [60], who provided gene family trees estimated from their nucleotide alignments (see Table 6.1 for an analysis of the number of copies per species in this dataset). In a prior study, Butler *et al.* [32] estimated species trees from this same dataset (specifically the concatenated alignment of putatively orthologous amino acid sequences) using MrBayes [203]. The comparison of trees estimated on this biological dataset is difficult to interpret, as the prior concatenation analysis was constrained to enforce the out-grouping of *S. castellii* with respect to *S. cerevisiae* and *C. glabrata*. Furthermore, the study by Butler *et al.* [32] reported several trees that differed with respect to this group (i.e., not all analyses returned this as a clade) as well as with respect to the placement of *K. waltii*. According to their study, none of these resolutions are clearly correct.

6.3.2 Simulated Datasets

Species trees and gene trees: We used SimPhy version 1.0.2 to simulate a collection of 100-species, 1000-gene datasets under the DLCoal model with six different model conditions: three levels of GDL and two levels of ILS. The easiest model condition was (largely) based on parameters estimated from the 16-species fungal dataset from Rasmussen and Kellis [60] except that we assumed 10 generations per year instead of $1.\bar{1}$ generations per year, which resulted in simulated datasets that were similar to the biological dataset in terms of each species being represented in a similar proportion of gene family trees (Tables 6.1–6.2); Supplementary Materials for details. To make more challenging model conditions, we increased the GDL rate and ILS level by increasing the effective population size (EPS).

We quantified the level of ILS by computing the normalized RF distance between each true locus tree and its respective true gene tree (which are on the same leaf set), averaging this value across all 1000 locus/gene trees. The average locus-to-gene tree discord across the 10 replicate datasets was 2% for the "no ILS condition" and 12% for the "low/moderate" ILS condition (note that if we had used 1.1 generations per year, AD would have increased to 19% and 55%, respectively).

We also quantified the level of GDL by counting the number of leaves and the number of species per gene tree. All gene trees had approximately 100 leaves, which is expected since the duplication and loss rates are equal. As the duplication/loss rate increased, the number of species per gene tree decreased, so even though locus/gene trees had the same number of leaves on average, these leaves were labeled by fewer species. For duplication/loss rates of 1×10^{-10}, 2×10^{-10}, and 5×10^{-10} the average number of species per gene tree was 85, 74, and 53.

We allowed gene trees to deviate from a strict molecular clock by using gene-by-lineage-specific rate heterogeneity modifiers, meaning that for each gene tree, a gamma distribution was defined for each gene tree by drawing α from a log-normal distribution with a location of 1.5 and a scale of one (same parameters as used in [261]), and then each branch length in a gene tree was multiplied by a value drawn the gamma distribution corresponding to that gene tree.

DNA (gene) sequence data: We now describe the simulation of DNA sequence data for each gene tree produced by SimPhy. Again, our protocol is also based on the fungal dataset from Rasmussen and Kellis [60], who provided an estimated multiple sequence alignment (MSA) and an estimated maximum likelihood (ML) tree for each of the 5 351 genes. We estimated GTR+GAMMA model parameters ($\vec{\pi}$, Q, and α) for each MSA and ML gene tree pair using RAxML version 8.2.12 and then fit distributions to the estimated GTR+GAMMA model parameters (only MSAs with least 500 parsimony-informative sites and at most 25% gaps were included in this analysis). For each gene tree, we drew GTR+GAMMA model parameters from these distributions and then simulated DNA sequences (with 1000 sites) using INDELible version 1.03. Because DNA sequences were simulated without insertions or deletions, MSA estimation was not necessary.

Estimated gene trees: ML gene trees were estimated under the GTR+GAMMA model using RAxML version 8.2.12. Prior to gene tree estimation, sequences were truncated to the first 25, 50, 100, and 250 nucleotides to produce datasets with varying levels of GTEE. GTEE was measured by the normalized RF distance between the true and the estimated gene family trees. Sequence lengths of 25, 50, 100, and 250 resulted in mean GTEE of 67%, 52%, 35%, and 19%, respectively.

6.3.3 Species Tree Estimation

Finally, species trees were estimated on 25, 50, 100, and 500 gene family trees, either true or estimated, using three different methods: ASTRAL-multi (as implemented in ASTRAL version 5.6.3), DupTree, FastMulRFS (as implemented in release 1.2.0 version 3), and MulRF (as implemented in version 2.1). All estimated species trees were binary (note that a single optimal tree was taken as an estimate of the species tree even when multiple equally optimal trees were returned by FastMulRFS). This created 120 model conditions (three GDL rates, two levels of ILS, five levels of GTEE, and four numbers of genes), each with 10 replicates, for a total of 1 200 datasets.

6.3.4 Evaluation

Species tree estimation methods were evaluated in terms of running time and species tree error, as measured by the RF error rate (Equation 2.3). All computational experiments were performed on the Campus Cluster at the University of Illinois at Urbana-Champaign, which is a heterogeneous system, meaning that compute nodes can have different specifications (https://campuscluster.illinois.edu/resources/docs/nodes/).

6.4 RESULTS

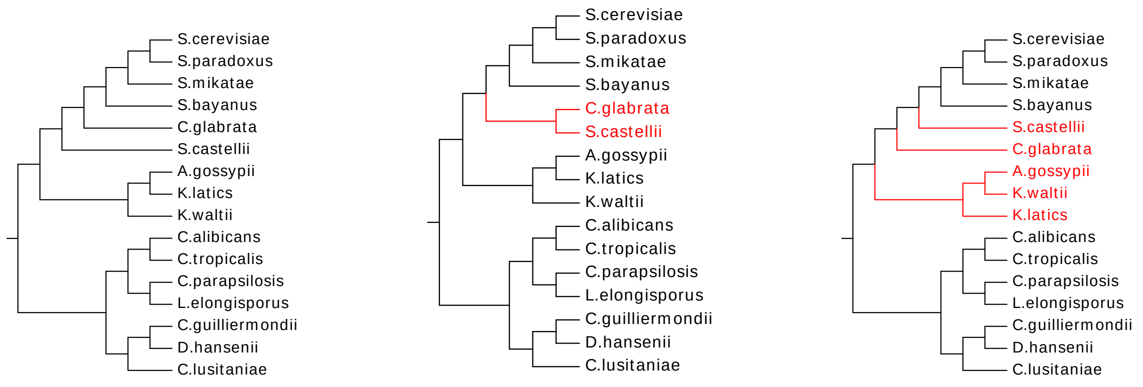
6.4.1 Fungal Dataset

In an analysis of the fungal dataset, all methods (ASTRAL-multi, FastMulRFS, DupTree, and MulRF) produced species trees that were similar to the MrBayes concatenation tree (Figure 6.3). The differences in species trees are minor given the variability in the trees found by Butler *et al.* [32], the use of a topological constraint in their MrBayes analysis, and the uncertainty about the placement of specific taxa in the tree (see Supplementary Information Section 5 in [32] for more information).

Given that the topological differences are minor and difficult to interpret, we focus on differences in empirical running time. FastMulRFS and DupTree completed in under a minute each, ASTRAL-multi completed in 18 minutes, and MulRF completed in 40 minutes. Hence, FastMulRFS is much faster than MulRF and ASTRAL-multi. While all four of these methods were relatively fast on 16 taxa, we expect the difference between methods to increase on datasets with larger numbers of species and with higher rates of gene duplication. The improvement in running time over MulRF and ASTRAL-multi is due in part to the fact that both MulRF and ASTRAL-multi use the original gene family trees, while FastMulRFS uses the reduced singly-labeled trees; hence, as the number of leaves or the duplication rate increase, the advantage in running time for FastMulRFS should also increase.

6.4.2 Simulated Datasets

DupTree typically had poorer accuracy than the other tested methods (Figures 6.4–reffig:fastmulrfs-100gen), especially when the level of GTEE was high. As high GTEE is consistent with the generally low bootstrap branch support values reported for several multi-locus datasets (Table 3.1), we focus on comparing MulRF, FastMulRFS, and ASTRAL-multi. All methods improved in accuracy with larger numbers of genes and degraded in



(a) Concatenation (b) ASTRAL-multi (c) FastMulRFS

Figure 6.3: Species trees were estimated on the 16-taxon fungal dataset with 5 351 gene family trees estimated by Rasmussen and Kellis [60]. Subfigure (a) shows the MrBayes concatenation tree estimated by Butler *et al.* [32], which is based on putative orthologs instead of the gene family trees. Subfigure (b) shows the ASTRAL-multi tree, and subfigure (c) shows the FastMulRFS tree, which is the same tree produced by MulRF and DupTree. Topological differences between the MrBayes concatenation tree are highlighted in red; however, this is not indicative of which resolution is correct or incorrect, as the true placement of these taxa is not yet established. DupTree and FastMulRFS were the fastest (both completed in less than a minute), ASTRAL-multi estimated a species tree in 18 minutes, and MulRF completed in 40 minutes.

accuracy with higher GTEE levels, ILS levels, and/or GDL rates. The relative accuracy between methods was consistent across all model conditions, although the degree of difference depended on the model conditions, with bigger differences for smaller numbers of genes and higher GTEE levels, ILS levels, and GDL rates. When given 500 gene trees, error levels were low and differences between methods were (usually) small, so that the main difference was running time. The fastest method was FastMulRFS, MulRF was the slowest, and ASTRAL-multi was intermediate.

FastMulRFS vs. MulRF: FastMulRFS and MulRF are both heuristics for the RFS-MUL-trees problem. They were essentially tied for accuracy across all tested conditions (Figures 6.4–6.5 and Table 6.3), but FastMulRFS was dramatically faster than MulRF (Figures 6.4–6.5 and Table 6.4). In addition, FastMulRFS nearly always returned a tree with an equivalent or better RFS-MUL-trees score than MulRF. Out of the 1 200 datasets analyzed, FastMulRFS was worse than MulRF on 56 datasets, FastMulRFS was equal to MulRF on

962 datasets, and FastMulRFS was better than MulRF on 182 datasets.

FastMulRFS vs. ASTRAL-multi: FastMulRFSwas always at least as accurate on average as ASTRAL-multi (and was often more accurate on average than ASTRAL-multi) across all model conditions tested (Figures 6.4–6.5 and Table 6.3), with larger differences between methods for the higher GTEE conditions and smaller differences for the lower GTEE conditions. The running times for ASTRAL-multi and FastMulRFS increased with the number of genes, but FastMulRFS was always faster (Figures 6.4–6.5 and Table 6.4). For example, on the 500-gene model conditions, FastMulRFS typically completed in one to two minutes (and always in under five minutes), whereas ASTRAL-multi used between 10 minutes and 1.2 hours.

6.5 DISCUSSION

To date, only two types of methods have been proven statistically consistent under any GDL model. The first type of method is based on maximizing quartets that are induced by the gene family trees, and the second type of method is based on maximizing bipartitions that are induced by the gene family trees. ASTRAL-multi is an example of the first kind of method, and FastMulRFS is an example of the second type of method; notably, both methods use DP to solve their optimization problems exactly within a constrained search space. The conditions under which these two methods have been proven statistically consistent are different. ASTRAL-multi is established to be consistent under a gene evolution model that requires that all the genes evolve *i.i.d.* within the species tree. Since the time of our study, Markin and Eulenstein [262] showed that a related quartet-based approach is statistically consistent under the DLCoal model, which allows for both GDL and ILS but still assumes that genes evolve *i.i.d.* within the species tree. In contrast, FastMulRFS has been proven consistent under a generic model that does not require the genes to evolve *i.i.d.* and indeed allows for a very generic model similar to the NCM model. This is a relative strength of FastMulRFS, as genes do not evolve *i.i.d.* within a species tree, as discussed in [263]. On the other hand, FastMulRFS has only been proven consistent when no adversarial GDL occurs and no ILS occurs; this is a relative weakness of FastMulRFS (although see Conjecture 6.1 regarding adversarial GDL). Therefore, from a theoretical perspective, there are advantages and disadvantages for both methods.

In terms of empirical performance, FastMulRFS was more accurate and more robust to GTEE than ASTRAL-multi under most model conditions we examined. The only conditions in which the two methods achieved similar accuracy were characterized by low GTEE and large numbers of genes, where both methods achieved very high accuracy. FastMulRFS also was much faster than ASTRAL-multi, with large improvements in speed for large numbers of genes and high GTEE. In summary, FastMulRFS had superior performance compared to ASTRAL-multi in our experimental evaluation.

A comparison between FastMulRFS and MulRF is also interesting. Both methods attempt to solve the same NP-hard optimization problem, and neither is guaranteed to find an optimal solution. However, FastMulRFS is guaranteed to find an optimal solution within a constrained search space in polynomial time; furthermore, the way that FastMulRFS constrains its search space is sufficient to ensure that it is statistically consistent. MulRF, in contrast, uses a locally optimal search strategy combined with hill climbing, so it does not run in polynomial time and may not be provably statistically consistent. In our experimental study, the two methods were very close in accuracy, but FastMulRFS was dramatically faster. Overall, FastMulRFS was superior to MulRF.

FastMulRFS matched or improved on the other methods under all conditions we explored, where gene trees evolved under a unified model of ILS and GDL (which did not prohibit adversarial GDL). Our study suggests that FastMulRFS may have good robustness and high accuracy, even under conditions where it has not (yet) been proven statistically consistent. (note that we think it is unlikely that FastMulRFS is statistically consistent under conditions with high ILS.) Future work is clearly needed to evaluate FastMulRFS and other methods under a wider range of model conditions, including explicit conditions where adversarial GDL and high levels of ILS occur. Simulations should also be performed to evaluate other scenarios that produce multi-copy genes, for example whole genome duplication events, which impact species tree estimation for many major clades, including fungi [32] and plants [25]. More complex simulations also should be performed (including gene conversion and HGT) in order to better understand the conditions in which methods perform well. Along these lines, it would be helpful to characterize biological datasets to understand realistic levels of ILS and GDL (including the frequency of adversarial GDL); this task has been identified as an important challenge throughout this dissertation.

A limitation of this study is that we examined only a few methods, and future studies should evaluate other methods, including *guenomu* (discussed earlier) and MixTreEM [264], to discover the places in the parameter space of model species trees where each method outperforms the others. Furthermore, methods that operate by making predictions of orthology could be used in a three-phase approach: given inputs with sequence alignments and MUL-trees, predict orthology, reduce to datasets with just orthologous genes (and hence singly-labeled gene trees), and then run a preferred species tree estimation method. For example, in a recent preprint, Zhang *et al.* [265] presented A-PRO, another modification of

ASTRAL, and proved that it is statistically consistent under a GDL model if given correctly "tagged" gene trees (i.e., each node in each gene tree is correctly identified as either a duplication or not); however, this assumption means that orthology can be inferred without error (an assumption that is not made for ASTRAL-multi). Studies evaluating A-PRO and these other approaches under a variety of conditions would enable biologists to select methods with the best expected accuracy for their datasets.

While we have focused on the estimated species topology, biologists require that branches in the topology to be annotated with information about the support for or confidence in each branch. Our reduction on MUL-trees (which transforms them into singly-labeled trees) is useful not only for estimating species trees but also for interpreting estimated species trees.

First, the reduction allows the branches of the species tree to be annotated by the proportion of MUL-trees that contain that branch; this metric is easy to interpret and therefore highly useful. Second, by combining the reduction with DP in a constrained search space, we can use an existing tool, called SIESTA [266], to quantify the space of optimal solutions; specifically, SIESTA reports the number of optimal solutions in the constrained search space and annotates each branch by the proportion of optimal solutions that contain that branch. This seems especially useful in the context of duplication and loss, as the reduction on MUL-trees illustrates that the MUL-trees are likely to be unresolved when duplications occur; this combined with losses could result in a large space of optimal solutions.

Third, after applying the reduction to MUL-trees, it possible to determine the relative weight of each MUL-tree when solving the RFS-MUL-trees problem. A MUL-tree M that evolves with only a duplication at a leaf induces $S(M) - 3$ valid bipartitions; however, a MUL-tree that evolves with only a duplication at the root induces zero valid bipartitions; therefore, the output tree could be biased towards a small subset of MUL-trees that contain a disproportionately large number of valid bipartitions compared to the other MUL-trees in the input set. While this is not a problem in theory, it can be a problem in practice, as MUL-trees can differ from the species tree due to estimation error and other sources of heterogeneity. This has already been shown to be an issue in the context of ILS. As an example, gene trees with different numbers of species induce different numbers of quartets, and in this case, each gene tree contributes a different number of quartets to the MQSS problem (which ASTRAL solves within a constrained search space). Two recent studies [267, 268] showed that the resolution of some controversial branches can change by removing outlier gene trees that contribute a disproportionately large number of quartets compared to the other gene trees in the input set. The ability to recognize that two fully resolved MUL-trees can contribute very different numbers of bipartitions to the RFS-MUL-trees problem is critical for identifying systematic biases, and we hope the theoretical observations of our study further enable the interpretation of species trees estimated from gene family trees.

6.6 CONCLUSIONS

We presented FastMulRFS, a new method for estimating species tree from unrooted gene family trees, without needing to have any information about orthology. FastMulRFS runs in polynomial-time and is provably statistically consistent under a generic GDL when adversarial GDL does not occur. Prior to our study, the only method established to be statistically consistent under any GDL model was ASTRAL-multi (their proof does assume that genes evolve *i.i.d.* within the species tree but does not prohibit adversarial GDL). Of course, statistical consistency does not predict performance on finite amounts of data or in the context of model violations (which are perhaps certain to occur when analyzing biological datasets). In our simulation study, FastMulRFS compared quite favorably (in terms of accuracy and running time) to ASTRAL-multi as well as MulRF. This is significant as we evaluated methods under conditions where neither FastMulRFS nor ASTRAL-multi are (yet) proven to be statistically consistent. Specifically, our proof establishes statistical consistency (of FastMulRFS) under models where ILS, GTEE, or adversarial GDL are prohibited, whereas our study benchmarks methods on datasets simulated with two ILS levels, five GTEE levels, and three GDL rates (note that adversarial GDL was not explicitly prohibited). Although accuracy is difficult to evaluate on biological datasets, FastMulRFS produced trees that were similar to those produced by other methods and did not violate known relationships. Overall, the recent advances in development of statistically consistent methods for species tree estimation under GDL models is exciting, and the good performance of many of these methods under a range of model conditions suggests that novel combinations and ideas may lead to even better methods that provide improved accuracy and scalability.

This section contains the four plots presented in Section 6.4 Results.

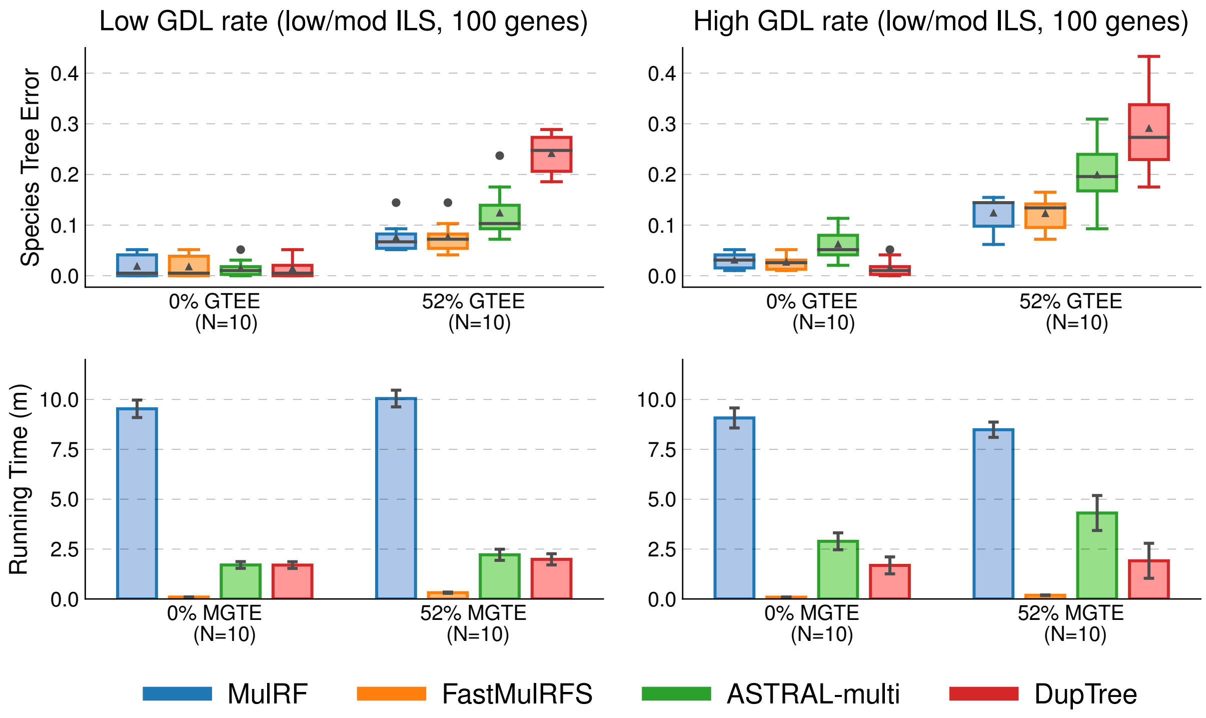


Figure 6.4: **Comparison of species tree estimation methods on 500 gene datasets with low/moderate ILS.** MulRF (blue), FastMulRFS (orange), ASTRAL-multi (green), and DupTree (red) are compared on 100-species, 500 gene datasets with low/moderate ILS (12%), two levels of GTEE, and two GDL rates (lowest and highest). Subplots in the top row show species tree error (RF error rate). Gray bars represent medians, gray triangles represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). Subplots in the bottom row show running time (in minutes); bars represent means and error bars represent standard deviations across replicate datasets.). The number N of replicates on which the methods completed is shown on the x-axis.

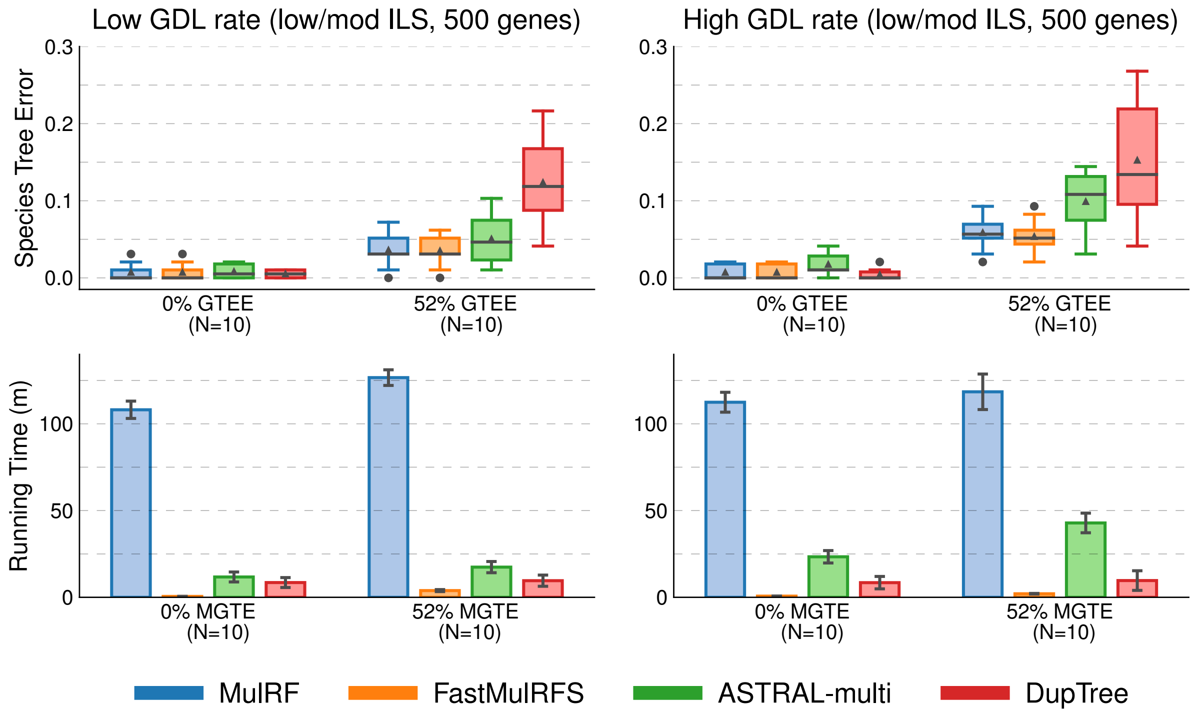


Figure 6.5: **Comparison of species tree estimation methods on 100 gene datasets with low/moderate ILS.** MulRF (blue), FastMulRFS (orange), ASTRAL-multi (green), and DupTree (red) are compared on 100-species, 100 gene datasets with low/moderate ILS (12%), two levels of GTEE, and two GDL rates (lowest and highest). Subplots in the top row show species tree error (RF error rate). Gray bars represent medians, gray triangles represent means, gray circles represent outliers, box plots are defined by quartiles (extending from the first to the third quartiles), and whiskers extend to plus/minus 1.5 times the interquartile distance (unless greater/less than the maximum/minimum value). Subplots in the bottom row show running time (in minutes); bars represent means and error bars represent standard deviations across replicate datasets.). The number N of replicates on which the methods completed is shown on the x-axis.

6.8 TABLES

This section contains the two tables presented in Section 6.3 Performance Study and the two tables presented in 6.4 Results.

Table 6.1: **Statistics on the number of copies of each species in the fungal datasets.** For the fungal biological dataset, we computed the mean (± standard deviation), the minimum, and the maximum number of copies of each species across 5 351 gene trees. We also report the proportion of gene family trees with more than 1, 2, 5, 10, and 20 copies of a species; for example, > 1 indicates the proportion of gene family trees (out of 5 351) with greater than one copy of the species.

Species	Mean ± Std	Min	Max	=1	>1	>2	>5	>10	>20
A. gossypii	0.85 ± 0.58	0	13	0.731	0.050	0.008	0.001	0.000	0.000
C. alibicans	1.04 ± 0.65	0	7	0.769	0.111	0.027	0.001	0.000	0.000
C. glabrata	0.93 ± 0.81	0	27	0.667	0.110	0.019	0.002	0.001	0.000
C. guilliermondii	0.99 ± 0.70	0	11	0.722	0.110	0.027	0.001	0.000	0.000
C. lusitaniae	0.95 ± 0.62	0	10	0.728	0.098	0.019	0.000	0.000	0.000
C. parapsilosis	1.00 ± 0.73	0	12	0.729	0.110	0.028	0.003	0.000	0.000
C. tropicalis	1.04 ± 0.73	0	8	0.738	0.122	0.033	0.003	0.000	0.000
D. hansenii	1.02 ± 0.65	0	7	0.756	0.110	0.026	0.002	0.000	0.000
K. latics	0.89 ± 0.63	0	15	0.745	0.063	0.010	0.002	0.000	0.000
K. waltii	0.88 ± 0.71	0	18	0.736	0.059	0.011	0.002	0.001	0.000
L. elongisporus	0.98 ± 0.66	0	9	0.727	0.108	0.021	0.001	0.000	0.000
S. bayanus	0.94 ± 0.81	0	23	0.643	0.128	0.022	0.002	0.000	0.000
S. castellii	1.01 ± 0.91	0	25	0.638	0.154	0.028	0.003	0.001	0.000
S. cerevisiae	1.03 ± 1.09	0	42	0.678	0.141	0.027	0.004	0.001	0.001
S. mikatae	0.92 ± 0.76	0	18	0.646	0.118	0.021	0.002	0.000	0.000
S. paradoxus	0.95 ± 0.83	0	25	0.649	0.129	0.023	0.002	0.000	0.000

Table 6.2: **Statistics on the number of copies of each species in the datasets simulated based on the fungal dataset.** This is the same as Table 6.1 but data for the first 15 species in the first replicate datasets are shown.

Species	Mean ± Std	Min	Max	=1	>1	>2	>5	>10	>20
Assuming 10 generations per year									
1	1.02 ± 0.57	0	4	0.743	0.128	0.021	0.000	0.000	0.000
2	1.01 ± 0.60	0	4	0.703	0.144	0.018	0.000	0.000	0.000
3	0.99 ± 0.56	0	5	0.736	0.119	0.012	0.000	0.000	0.000
4	1.01 ± 0.60	0	4	0.703	0.142	0.019	0.000	0.000	0.000
5	1.01 ± 0.61	0	4	0.707	0.140	0.024	0.000	0.000	0.000
6	1.02 ± 0.57	0	4	0.748	0.124	0.021	0.000	0.000	0.000
7	1.05 ± 0.57	0	3	0.726	0.151	0.019	0.000	0.000	0.000
8	1.02 ± 0.58	0	4	0.747	0.125	0.020	0.000	0.000	0.000
9	1.01 ± 0.57	0	4	0.738	0.126	0.019	0.000	0.000	0.000
10	1.02 ± 0.56	0	4	0.759	0.121	0.019	0.000	0.000	0.000
11	1.02 ± 0.58	0	4	0.742	0.127	0.023	0.000	0.000	0.000
12	0.99 ± 0.56	0	5	0.743	0.114	0.013	0.000	0.000	0.000
13	0.96 ± 0.59	0	4	0.709	0.119	0.016	0.000	0.000	0.000
14	1.01 ± 0.60	0	4	0.732	0.124	0.025	0.000	0.000	0.000
15	1.01 ± 0.63	0	4	0.693	0.145	0.022	0.000	0.000	0.000
Assuming $1.\bar{1}$ *generations per year*									
1	0.99 ± 0.20	0	3	0.961	0.015	0.001	0.000	0.000	0.000
2	0.99 ± 0.21	0	2	0.955	0.017	0.000	0.000	0.000	0.000
3	0.99 ± 0.18	0	2	0.968	0.012	0.000	0.000	0.000	0.000
4	1.00 ± 0.22	0	3	0.954	0.021	0.001	0.000	0.000	0.000
5	1.00 ± 0.22	0	2	0.950	0.024	0.000	0.000	0.000	0.000
6	0.99 ± 0.19	0	3	0.965	0.011	0.001	0.000	0.000	0.000
7	0.99 ± 0.17	0	2	0.971	0.009	0.000	0.000	0.000	0.000
8	0.99 ± 0.19	0	3	0.965	0.013	0.001	0.000	0.000	0.000
9	0.99 ± 0.19	0	3	0.966	0.011	0.001	0.000	0.000	0.000
10	0.99 ± 0.20	0	3	0.963	0.013	0.001	0.000	0.000	0.000
11	1.00 ± 0.20	0	3	0.962	0.016	0.001	0.000	0.000	0.000
12	0.99 ± 0.18	0	2	0.969	0.012	0.000	0.000	0.000	0.000
13	1.00 ± 0.18	0	2	0.969	0.016	0.000	0.000	0.000	0.000
14	0.99 ± 0.18	0	2	0.966	0.014	0.000	0.000	0.000	0.000
15	1.00 ± 0.20	0	3	0.963	0.017	0.001	0.000	0.000	0.000

Table 6.3: **Species tree error on the simulated datasets.** Species tree error is averaged across 10 replicate datasets for each of the model conditions on 100-species simulated datasets for **ASTRAL-multi / FastMulRFS / MulRF**. ILS level is measured using the average normalized RF distance between true locus trees and true gene trees.

GTEE	25 genes	50 genes	100 genes	500 genes
ILS: 2%, D/L Rate: 1E-10				
67%	0.26 / 0.21 / 0.23	0.19 / 0.12 / 0.13	0.17 / 0.08 / 0.09	0.10 / 0.04 / 0.05
52%	0.14 / 0.11 / 0.11	0.09 / 0.07 / 0.07	0.07 / 0.05 / 0.05	0.03 / 0.02 / 0.02
35%	0.07 / 0.06 / 0.06	0.05 / 0.05 / 0.05	0.04 / 0.03 / 0.03	0.02 / 0.02 / 0.02
19%	0.04 / 0.04 / 0.04	0.03 / 0.03 / 0.02	0.02 / 0.02 / 0.02	0.01 / 0.01 / 0.01
0%	0.01 / 0.01 / 0.01	0.01 / 0.00 / 0.00	0.00 / 0.00 / 0.00	0.00 / 0.00 / 0.00
ILS: 2%, D/L Rate: 2E-10				
67%	0.33 / 0.29 / 0.36	0.27 / 0.18 / 0.21	0.19 / 0.14 / 0.16	0.12 / 0.06 / 0.07
52%	0.19 / 0.16 / 0.16	0.15 / 0.10 / 0.10	0.11 / 0.08 / 0.08	0.05 / 0.04 / 0.04
35%	0.11 / 0.09 / 0.09	0.08 / 0.05 / 0.05	0.05 / 0.04 / 0.04	0.03 / 0.02 / 0.02
19%	0.07 / 0.05 / 0.05	0.04 / 0.03 / 0.03	0.03 / 0.03 / 0.03	0.01 / 0.01 / 0.01
0%	0.01 / 0.01 / 0.01	0.01 / 0.00 / 0.00	0.01 / 0.00 / 0.00	0.01 / 0.00 / 0.00
ILS: 2%, D/L Rate: 5E-10				
67%	0.48 / 0.42 / 0.48	0.37 / 0.29 / 0.33	0.32 / 0.19 / 0.22	0.19 / 0.06 / 0.07
52%	0.33 / 0.27 / 0.29	0.23 / 0.17 / 0.17	0.18 / 0.10 / 0.10	0.09 / 0.04 / 0.04
35%	0.19 / 0.17 / 0.16	0.14 / 0.10 / 0.09	0.11 / 0.06 / 0.06	0.05 / 0.02 / 0.02
19%	0.11 / 0.09 / 0.08	0.09 / 0.05 / 0.05	0.07 / 0.03 / 0.03	0.02 / 0.02 / 0.02
0%	0.04 / 0.01 / 0.01	0.04 / 0.01 / 0.01	0.02 / 0.01 / 0.01	0.00 / 0.00 / 0.00
ILS: 12%, D/L Rate: 1E-10				
67%	0.32 / 0.24 / 0.27	0.24 / 0.15 / 0.17	0.18 / 0.11 / 0.14	0.11 / 0.06 / 0.06
52%	0.19 / 0.14 / 0.15	0.13 / 0.12 / 0.11	0.12 / 0.08 / 0.08	0.05 / 0.04 / 0.04
35%	0.11 / 0.09 / 0.09	0.07 / 0.06 / 0.06	0.05 / 0.05 / 0.05	0.03 / 0.02 / 0.02
19%	0.07 / 0.06 / 0.06	0.05 / 0.04 / 0.04	0.03 / 0.03 / 0.03	0.02 / 0.01 / 0.01
0%	0.04 / 0.03 / 0.04	0.02 / 0.02 / 0.02	0.01 / 0.02 / 0.02	0.01 / 0.01 / 0.01
ILS: 12%, D/L Rate: 2E-10				
67%	0.35 / 0.29 / 0.34	0.26 / 0.21 / 0.23	0.20 / 0.14 / 0.15	0.11 / 0.07 / 0.07
52%	0.19 / 0.16 / 0.16	0.15 / 0.12 / 0.12	0.12 / 0.08 / 0.07	0.06 / 0.03 / 0.03
35%	0.12 / 0.10 / 0.10	0.09 / 0.09 / 0.08	0.08 / 0.05 / 0.05	0.03 / 0.02 / 0.02
19%	0.07 / 0.06 / 0.06	0.05 / 0.05 / 0.05	0.05 / 0.03 / 0.03	0.02 / 0.02 / 0.02
0%	0.04 / 0.03 / 0.03	0.03 / 0.03 / 0.02	0.02 / 0.01 / 0.01	0.01 / 0.01 / 0.01
ILS: 12%, D/L Rate: 5E-10				
67%	0.47 / 0.43 / 0.50	0.41 / 0.28 / 0.33	0.31 / 0.20 / 0.22	0.17 / 0.09 / 0.09
52%	0.36 / 0.28 / 0.32	0.27 / 0.17 / 0.19	0.20 / 0.12 / 0.12	0.10 / 0.05 / 0.06
35%	0.24 / 0.17 / 0.18	0.17 / 0.10 / 0.10	0.13 / 0.08 / 0.08	0.06 / 0.03 / 0.03
19%	0.18 / 0.12 / 0.11	0.13 / 0.08 / 0.08	0.10 / 0.06 / 0.06	0.04 / 0.03 / 0.03
0%	0.12 / 0.05 / 0.05	0.08 / 0.04 / 0.04	0.06 / 0.03 / 0.03	0.02 / 0.01 / 0.01

Table 6.4: **Running time on the simulated datasets.** Running time (in seconds) is averaged across 10 replicate datasets on 100-species simulated datasets for each of the model conditions for **ASTRAL-multi / FastMulRFS / MulRF**. ILS level is measured using the average normalized RF distance between true locus trees and true gene trees.

GTEE	25 genes	50 genes	100 genes	500 genes
ILS: 2%, D/L Rate: 1E-10				
67%	42 / 5 / 113	102 / 9 / 263	245 / 22 / 614	2148 / 357 / 7089
52%	28 / 4 / 95	61 / 6 / 262	156 / 14 / 565	1181 / 306 / 7225
35%	22 / 3 / 89	51 / 5 / 211	121 / 11 / 623	756 / 82 / 6921
19%	19 / 2 / 86	37 / 3 / 226	86 / 7 / 596	641 / 50 / 6727
0%	20 / 2 / 99	46 / 2 / 228	106 / 4 / 582	769 / 24 / 6431
ILS: 2%, D/L Rate: 2E-10				
67%	52 / 4 / 91	123 / 8 / 273	306 / 20 / 580	2514 / 273 / 6937
52%	36 / 4 / 99	76 / 6 / 280	213 / 12 / 577	1357 / 167 / 7225
35%	24 / 3 / 92	49 / 5 / 254	173 / 9 / 636	930 / 90 / 6987
19%	26 / 2 / 105	47 / 4 / 237	159 / 7 / 613	978 / 50 / 6852
0%	26 / 2 / 103	55 / 3 / 197	157 / 5 / 597	829 / 30 / 6827
ILS: 2%, D/L Rate: 5E-10				
67%	86 / 4 / 102	232 / 7 / 232	418 / 16 / 585	4495 / 201 / 6884
52%	49 / 3 / 114	128 / 5 / 691	258 / 11 / 692	2615 / 125 / 6793
35%	40 / 3 / 112	79 / 4 / 265	195 / 8 / 565	1856 / 68 / 6635
19%	32 / 2 / 111	84 / 3 / 241	177 / 6 / 546	1288 / 53 / 6645
0%	36 / 2 / 110	83 / 3 / 215	153 / 5 / 542	1308 / 35 / 6608
ILS: 12%, D/L Rate: 1E-10				
67%	43 / 5 / 112	93 / 10 / 239	214 / 28 / 680	2107 / 394 / 6882
52%	29 / 4 / 111	63 / 7 / 573	133 / 19 / 603	1044 / 232 / 7597
35%	21 / 3 / 105	55 / 6 / 479	125 / 12 / 643	726 / 144 / 7351
19%	22 / 2 / 77	49 / 4 / 232	111 / 8 / 610	673 / 56 / 6976
0%	21 / 2 / 83	44 / 3 / 318	102 / 6 / 572	704 / 31 / 6485
ILS: 12%, D/L Rate: 2E-10				
67%	57 / 4 / 109	130 / 9 / 510	302 / 22 / 694	3098 / 292 / 7582
52%	32 / 4 / 107	74 / 7 / 383	193 / 15 / 583	1767 / 174 / 7085
35%	28 / 3 / 103	62 / 5 / 223	166 / 10 / 672	1052 / 95 / 6896
19%	22 / 2 / 106	53 / 4 / 238	182 / 7 / 541	836 / 54 / 7189
0%	23 / 2 / 104	47 / 3 / 279	121 / 6 / 586	808 / 35 / 6669
ILS: 12%, D/L Rate: 5E-10				
67%	105 / 4 / 109	205 / 7 / 431	469 / 16 / 599	4368 / 196 / 6889
52%	63 / 3 / 128	115 / 5 / 274	258 / 11 / 509	2572 / 123 / 7107
35%	46 / 3 / 112	101 / 4 / 204	185 / 8 / 647	1896 / 64 / 6745
19%	37 / 2 / 112	82 / 4 / 261	193 / 7 / 638	1392 / 54 / 6786
0%	37 / 2 / 95	73 / 3 / 251	173 / 5 / 544	1402 / 41 / 6746

6.9 ALGORITHMS

This section contains the algorithm presented in Section 6.2 Approach.

Algorithm 6.2: Preprocess MUL-trees.

Input : Unrooted, fully resolved MUL-tree $T = (t, S, \phi)$ with $|L(t)| = m$ and $|S| = n$ (Note that unrooted means there is a trifurcation at the "root" node.)
Output: $\mathcal{R}(\mathcal{X}(T))$

```
Function PreprocessMulTree($T$):
    $below \leftarrow [0]_{2m\times n}$; $above \leftarrow [0]_{2m\times n}$; $found \leftarrow [0]_n$;

    Step 1: Fill vector with 1 if species is below node and with 0 otherwise.
    for $v \in$ PostOrderNodeTraversal($T$) do
        if IsLeaf($v$) then
            $below[v, \phi(v)] \leftarrow 1$
        else if IsRoot($v$) then
            $[l, m, r] \leftarrow$ GetChildren($v$)
            for $s \in S$ do
                $below[v, s] \leftarrow below[l, s] \vee below[m, s] \vee below[r, s]$
        else
            $[l, r] \leftarrow$ GetChildren($v$)
            for $s \in S$ do
                $below[v, s] \leftarrow below[l, s] \vee below[r, s]$

    Step 2: Fill vector with 1 if species is "above" node and with 0 otherwise.
    $root \leftarrow$ GetRoot($T$)
    $[l, m, r] \leftarrow$ GetChildren($root$)
    for $s \in S$ do
        $above[l][s] \leftarrow below[m][s] \vee below[r][s]$
        $above[m][s] \leftarrow below[l][s] \vee below[r][s]$
        $above[r][s] \leftarrow below[l][s] \vee below[m][s]$
    for $v \in$ PreOrderNodeTraversal($T$) do
        SetEdgeLength($(v, p)$, 1)
        if $v \notin \{root, l, m, r\}$ and not IsLeaf($v$) then
            $p \leftarrow$ GetParent($v$)
            $[x, y] \leftarrow$ GetChildren(p)
            if $v == x$ then  $x \leftarrow$ y
            for $s \in S$ do
                $above[v][s] \leftarrow above[p][s] \vee below[x][s]$
                if $below[v][s] \wedge above[v][s]$ then
                    SetEdgeLength($(v, p)$, 0)

    Steps 3 and 4: Contract internal edges that fail to induce bipartitions and then remove extra
    copies of species.
    ContractEdgesWithZeroLength($T$)
    for $l \in L(t)$ do
        if $found[\phi(l)]$ then PruneLeaf($l$)
        else  $found[\phi(l)] \leftarrow 1$

    return $T$
```

CHAPTER 7: CONCLUSION

In this dissertation, we presented three new supertree-like methods, NJMerge, TreeMerge, and FastMulRFS, that can be used in the context of species tree estimation from genome-scale datasets.

The first two methods, NJMerge and TreeMerge, deviate from traditional supertree methods by taking a set of leaf-label-disjoint trees as input; thus we refer to these methods as disjoint tree mergers (DTMs). This specification implies that a compatibility supertree for the input trees exists—but also that the input trees provide no information on how to recover a compatibility supertree that is close to the true phylogeny. As such, DTM methods require auxiliary data as input (note that they estimate the phylogeny from this data subject to the topological constraints defined by the input trees). Of the DTM methods, TreeMerge is particularly noteworthy, as it seeks to reduce computational and storage requirements (and improve accuracy) by exploiting locality, combining an embarrassingly parallel local merge phase with a fast global merge phase. We explore the use of DTM methods in divide-and-conquer species tree estimation pipelines, providing proofs of statistical consistency under the Multi-Species Coalescent (MSC) model and presenting empirical results for simulated datasets (on which the DTM-based pipelines compare favorably in terms of accuracy and running time to the dominant species tree estimation methods).

The third method, FastMulRFS, deviates from traditional supertree methods by taking as input a set of multi-labeled trees (MUL-trees). We show the input MUL-trees can be reduced to set of singly-labeled trees prior to solving FastMulRF's optimization problem: the Robinson-Foulds supertree (RFS) problem for MUL-trees. After applying this transformation, we then utilize a dynamic programming (DP) technique for solving the problem exactly within a constrained search space defined by the input. We explore the use of FastMulRFS in the context of species tree estimation from gene family trees, providing proofs of statistical consistency under a generic gene duplication and loss (GDL) model (provided no adversarial GDL occurs) and presenting empirical results for a fungal dataset and for simulated datasets (on which FastMulRFS compares favorably in terms of accuracy and running time to other leading methods that take MUL-trees as input).

All three of these methods are combinatorial in nature and operate, at least in part, by constraining the space of allowed solutions. The goal is to improve scalability by constraining the solution space but to do so in such a way that good empirical properties (accuracy in simulation studies) and good theoretical properties (statistical consistency) are maintained. One of the major challenges is that the (relative) performance of methods and the (relative) benefit of using different techniques for improving scalability depends on the model condition. This requires method developers to perform extensive simulations, benchmarking methods on a wide range of model conditions. Even with these efforts, interpreting the results can be difficult when it is unclear which of the model conditions best reflects reality for a given dataset or for many datasets. This is largely an unknown, as it is still difficult to quantify "model conditions" in biological datasets due to model violations, estimation error (from data preprocessing), and missing data. These issues complicate experimental design and method development.

We have outlined potential future work throughout this dissertation and now review several broad avenues for method developers.

First, we have consistently identified gene tree estimation error (GTEE) as a potential problem for gene tree summary methods. While summary methods are surprisingly robust to GTEE in simulation studies, this may not be the case in biological studies when estimation error may be systematically biased. Developing methods to detect estimation error, especially systematic error, and developing methods that can leverage large numbers of loci, where each locus has low phylogenetic signal, are important areas of future research.

Second, we have considered theoretical guarantees under the MSC model or a generic GDL model, but not under both. Multiple sources of gene tree heterogeneity can occur in practice, and examining statistical guarantees and empirical performance within such regimes is an important area of future research (e.g., [262]). In particular, it may be important to account for sources of heterogeneity, such as horizontal gene transfer (HGT) in bacteria and hybridization in eukaryotes, that must be modeled by a species network rather than a species tree (e.g., [269]). Phylogenetic network estimation is far more computationally intensive than tree estimation [95], so this is an area where method development, especially focusing on scalability to large datasets, could be highly impactful.

Third, there is growing interest in estimating phylogenies under parameter-rich statistical models (e.g., [248]) using maximum likelihood (ML) methods [118] or Bayesian methods (e.g., [42, 43, 203, 218, 270]). Recent work by Wang *et al.* [271] suggests that constraints can be applied in the context of Bayesian phylogenetic network estimation to improve computational performance; the combination of constrained estimation with more biologically realistic models and/or Bayesian methods is largely unexplored to date. Of particular interest to us is how locality in the solution space can be efficiently detected and then exploited (e.g., through the use of constraints) to reduce running time and to improve parallelism. Machine learning and dynamic control algorithms could be useful in this context.

These computational and statistical challenges together with the large-scale sequencing efforts currently underway make it an exciting time to be working in phylogenomics.

CHAPTER 8: REFERENCES

[1] A. P. Dhanapal and M. Govindaraj, “International Unlimited Thirst for Genome Sequencing, Data Interpretation, and Database Usage in Genomic Era: The Road towards Fast-Track Crop Plant Improvement,” *Genome Research*, vol. 2015, p. 684321, 2015.

[2] R. Levine, “i5k: The 5,000 Insect Genome Project,” *American Entomologist*, vol. 57, no. 2, pp. 110–113, 2011.

[3] i5K Consortium, “The i5K Initiative: Advancing Arthropod Genomics for Knowledge, Human Health, Agriculture, and the Environment,” *Journal of Heredity*, vol. 104, no. 5, pp. 595–600, 2013.

[4] S. Cheng, M. a. Melkonian, S. A. Smith, S. Brockington, J. M. Archibald, P.-M. Delaux, F.-W. Li, B. Melkonian, E. V. Mavrodiev, W. Sun, Y. Fu, H. Yang, D. E. Soltis, S. W. Graham, P. S. Soltis, X. Liu, X. Xu, and G. K.-S. Wong, “10KP: A phylodiverse genome sequencing plan,” *GigaScience*, vol. 7, no. 3, 2018.

[5] “Vertebrate genomes project,” February 2020. [Online]. Available: https://vertebrategenomesproject.org/

[6] H. A. Lewin, G. E. Robinson, W. J. Kress, W. J. Baker, J. Coddington, K. A. Crandall, R. Durbin, S. V. Edwards, F. Forest, M. T. P. Gilbert, M. M. Goldstein, I. V. Grigoriev, K. J. Hackett, D. Haussler, E. D. Jarvis, W. E. Johnson, A. Patrinos, S. Richards, J. C. Castilla-Rubio, M.-A. van Sluys, P. S. Soltis, X. Xu, H. Yang, and G. Zhang, “Earth BioGenome Project: Sequencing life for the future of life,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 115, no. 17, pp. 4325–4333, 2018.

[7] T. H. Jukes and C. R. Cantor, “Evolution of protein molecules,” in *Mammalian Protein Metabolism*, H. N. Munro, Ed. New York, NY, USA: Academic Press, 1969, vol. 3, pp. 21–132.

[8] M. Kimura, “A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences,” *Journal of Molecular Evolution*, vol. 16, pp. 111–120, 1980.

[9] S. Tavaré, “Some probabilistic and statistical problems in the analysis of DNA sequences,” *Lectures on mathematics in the life sciences*, vol. 17, no. 2, pp. 57–86, 1986.

[10] J. Felsenstein, “Evolutionary trees from DNA sequences: A maximum likelihood approach,” *Journal of Molecular Evolution*, vol. 17, no. 6, pp. 368–376, 1981.

[11] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, “Multiple sequence alignment modeling: methods and applications,” *Briefings in Bioinformatics*, vol. 17, no. 6, pp. 1009–1023, 2015.

[12] T. Warnow, *Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation.* Cambridge, United Kingdom: Cambridge University Press, 2017.

[13] J. A. Eisen, "Phylogenomics: Improving Funcational Predictions for Uncharacterized Genes by Evolutionary Analysis," *Genome Research*, vol. 8, no. 3, pp. 163–167, 1998.

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

[15] M. Syvanen, "Cross-species gene transfer; implications for a new theory of evolution," *Journal of Theoretical Biology*, vol. 112, pp. 333–343, 1985.

[16] W. Maddison, "Gene Trees in Species Trees," *Systematic Biology*, vol. 46, no. 3, pp. 523–536, 1997.

[17] J. C. Avise, J. F. Shapira, S. W. Daniel, C. F. Aquadro, and R. A. Lansman, "Mitochondrial DNA differentiation during the speciation process in Peromyscus." *Molecular Biology and Evolution*, vol. 1, no. 1, pp. 38–56, 1983.

[18] P. Pamilo and M. Nei, "Relationships between gene trees and species trees," *Molecular Biology and Evolution*, vol. 5, no. 5, pp. 568–583, 1988.

[19] N. Takahata, "Gene genealogy in three related populations: consistency probability between gene and population trees," *Genetics*, vol. 122, no. 4, pp. 967–966, 1989.

[20] F. Tajima, "Evolutionary relationship of DNA sequences in finite populations," *Genetics*, vol. 105, no. 2, pp. 437–460, 1983.

[21] N. A. Rosenberg, "The probability of topological concordance of gene trees and species trees," *Theoretical Population Biology*, vol. 61, no. 2, pp. 225–247, 2002.

[22] B. Rannala and Z. Yang, "Bayes Estimation of Species Divergence Times and Ancestral Population Sizes Using DNA Sequences From Multiple Loci," *Genetics*, vol. 164, no. 4, pp. 1645–1656, 2003.

[23] E. D. Jarvis, S. Mirarab, A. J. Aberer, B. Li, P. Houde, C. Li, S. Y. W. Ho, B. C. Faircloth, B. Nabholz, J. T. Howard, A. Suh, C. C. Weber, R. R. da Fonseca, J. Li, F. Zhang, H. Li, L. Zhou, N. Narula, L. Liu, G. Ganapathy, B. Boussau, M. S. Bayzid, V. Zavidovych, S. Subramanian, T. Gabaldón, S. Capella-Gutiérrez, J. Huerta-Cepas, B. Rekepalli, K. Munch, M. Schierup, B. Lindow, W. C. Warren, D. Ray, R. E. Green, M. W. Bruford, X. Zhan, A. Dixon, S. Li, N. Li, Y. Huang, E. P. Derryberry, M. F. Bertelsen, F. H. Sheldon, R. T. Brumfield, C. V. Mello, P. V. Lovell, M. Wirthlin, M. P. C. Schneider, F. Prosdocimi, J. A. Samaniego, A. M. V. Velazquez, A. Alfaro-Núñez, P. F. Campos, B. Petersen, T. Sicheritz-Ponten, A. Pas, T. Bailey, P. Scofield, M. Bunce, D. M. Lambert, Q. Zhou, P. Perelman, A. C. Driskell, B. Shapiro, Z. Xiong, Y. Zeng, S. Liu, Z. Li, B. Liu, K. Wu, J. Xiao, X. Yinqi, Q. Zheng, Y. Zhang, H. Yang, J. Wang, L. Smeds, F. E. Rheindt, M. Braun, J. Fjeldsa, L. Orlando, F. K. Barker, K. A. Jønsson, W. Johnson, K.-P. Koepfli, S. O'Brien, D. Haussler, O. A. Ryder, C. Rahbek, E. Willerslev, G. R. Graves, T. C. Glenn, J. McCormack, D. Burt, H. Ellegren, P. Alström, S. V. Edwards, A. Stamatakis, D. P. Mindell, J. Cracraft, E. L. Braun, T. Warnow, W. Jun, M. T. P. Gilbert, and G. Zhang, "Whole-genome analyses resolve early branches in the tree of life of modern birds," *Science*, vol. 346, no. 6215, pp. 1320–1331, 2014.

[24] N. J. Wickett, S. Mirarab, N. Nguyen, T. Warnow, E. Carpenter, N. Matasci, S. Ayyampalayam, M. S. Barker, J. G. Burleigh, M. A. Gitzendanner, B. R. Ruhfel, E. Wafula, J. P. Der, S. W. Graham, S. Mathews, M. Melkonian, D. E. Soltis, P. S. Soltis, N. W. Miles, C. J. Rothfels, L. Pokorny, A. J. Shaw, L. DeGironimo, D. W. Stevenson, B. Surek, J. C. Villarreal, B. Roure, H. Philippe, C. W. dePamphilis, T. Chen, M. K. Deyholos, R. S. Baucom, T. M. Kutchan, M. M. Augustin, J. Wang, Y. Zhang, Z. Tian, Z. Yan, X. Wu, X. Sun, G. K.-S. Wong, and J. Leebens-Mack, "Phylotranscriptomic analysis of the origin and early diversification of land plants," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 45, pp. E4859–E4868, 2014.

[25] J. H. Leebens-Mack, M. S. Barker, E. J. Carpenter, M. K. Deyholos, M. A. Gitzendanner, S. W. Graham, I. Grosse, Z. Li, M. Melkonian, S. Mirarab, M. Porsch, M. Quint, S. A. Rensing, D. E. Soltis, P. S. Soltis, D. W. Stevenson, K. K. Ullrich, N. J. Wickett, L. DeGironimo, P. P. Edger, I. E. Jordon-Thaden, S. Joya, T. Liu, B. Melkonian, N. W. Miles, L. Pokorny, C. Quigley, P. Thomas, J. C. Villarreal, M. M. Augustin, M. D. Barrett, R. S. Baucom, D. J. Beerling, R. M. Benstein, E. Biffin, S. F. Brockington, D. O. Burge, J. N. Burris, K. P. Burris, V. Burtet-Sarramegna, A. L. Caicedo, S. B. Cannon, Z. Çebi, Y. Chang, C. Chater, J. M. Cheeseman, T. Chen, N. D. Clarke, H. Clayton, S. Covshoff, B. J. Crandall-Stotler, H. Cross, C. W. dePamphilis, J. P. Der, R. Determann, R. C. Dickson, V. S. Di Stilio, S. Ellis, E. Fast, N. Feja, K. J. Field, D. A. Filatov, P. M. Finnegan, S. K. Floyd, B. Fogliani, N. García, G. Gâteblé, G. T. Godden, F. Q. Y. Goh, S. Greiner, A. Harkess, J. M. Heaney, K. E. Helliwell, K. Heyduk, J. M. Hibberd, R. G. J. Hodel, P. M. Hollingsworth, M. T. J. Johnson, R. Jost, B. Joyce, M. V. Kapralov, E. Kazamia, E. A. Kellogg, M. A. Koch, M. Von Konrat, K. Könyves, T. M. Kutchan, V. Lam, A. Larsson, A. R. Leitch, R. Lentz, F.-W. Li, A. J. Lowe, M. Ludwig, P. S. Manos, E. Mavrodiev, M. K. McCormick, M. McKain, T. McLellan, J. R. McNeal, R. E. Miller, M. N. Nelson, Y. Peng, P. Ralph, D. Real, C. W. Riggins, M. Ruhsam, R. F. Sage, A. K. Sakai, M. Scascitella, E. E. Schilling, E.-M. Schlösser, H. Sederoff, S. Servick, E. B. Sessa, A. J. Shaw, S. W. Shaw, E. M. Sigel, C. Skema, A. G. Smith, A. Smithson, C. N. Stewart, J. R. Stinchcombe, P. Szövényi, J. A. Tate, H. Tiebel, D. Trapnell, M. Villegente, C.-N. Wang, S. G. Weller, M. Wenzel, S. Weststrand, J. H. Westwood, D. F. Whigham, S. Wu, A. S. Wulff, Y. Yang, D. Zhu, C. Zhuang, J. Zuidof, M. W. Chase, J. C. Pires, C. J. Rothfels, J. Yu, C. Chen, L. Chen, S. Cheng, J. Li, R. Li, X. Li, H. Lu, Y. Ou, X. Sun, X. Tan, J. Tang, Z. Tian, F. Wang, J. Wang, X. Wei, X. Xu, Z. Yan, F. Yang, X. Zhong, F. Zhou, Y. Zhu, Y. Zhang, S. Ayyampalayam, T. J. Barkman, N. Nguyen, N. Matasci, D. R. Nelson, E. Sayyari, E. K. Wafula, R. L. Walls, T. Warnow, H. An, N. Arrigo, A. E. Baniaga, S. Galuska, S. A. Jorgensen, T. I. Kidder, H. Kong, P. Lu-Irving, H. E. Marx, X. Qi, C. R. Reardon, B. L. Sutherland, G. P. Tiley, S. R. Welles, R. Yu, S. Zhan, L. Gramzow, G. Theißen, G. K.-S. Wong, and O. T. P. T. Initiative, "One thousand plant transcriptomes and the phylogenomics of green plants," *Nature*, vol. 574, no. 7780, pp. 679–685, 2019.

[26] C. W. Linkem, V. N. Minin, and A. D. Leaché, "Detecting the Anomaly Zone in Species Trees and Evidence for a Misleading Signal in Higher-Level Skink Phylogeny (Squamata: Scincidae)," *Systematic Biology*, vol. 65, no. 3, pp. 465–477, 2016.

[27] J. E. McCormack, B. C. Faircloth, N. G. Crawford, P. A. Gowaty, R. T. Brumfield, and T. C. Glenn, "Ultraconserved elements are novel phylogenomic markers that resolve placental mammal phylogeny when combined with species-tree analysis," *Genome Research*, vol. 22, no. 4, pp. 746–754, 2012.

[28] J. H. Degnan and N. A. Rosenberg, "Gene tree discordance, phylogenetic inference and the multispecies coalescent," *Trends in Ecology and Evolution*, vol. 24, no. 6, pp. 332–340, 2009.

[29] S. V. Edwards, "Is a new and general theory of molecular systematics emerging?" *Evolution*, vol. 63, no. 1, pp. 1–19, 2009.

[30] S. V. Edwards, Z. Xi, A. Janke, B. C. Faircloth, J. E. McCormack, T. C. Glenn, B. Zhong, S. Wu, E. M. Lemmon, A. R. Lemmon, A. D. Leaché, L. Liu, and C. C. Davis, "Implementing and testing the multispecies coalescent model: A valuable paradigm for phylogenomics," *Molecular Phylogenetics and Evolution*, vol. 94, pp. 447–462, 2016.

[31] L. Arvestad, A.-C. Berglund, J. Lagergren, and B. Sennblad, "Bayesian gene/species tree reconciliation and orthology analysis using MCMC," *Bioinformatics*, vol. 19, no. Suppl 1, pp. i7–i15, 2003.

[32] G. Butler, M. D. Rasmussen, M. F. Lin, M. A. S. Santos, S. Sakthikumar, C. A. Munro, E. Rheinbay, M. Grabherr, A. Forche, J. L. Reedy, I. Agrafioti, M. B. Arnaud, S. Bates, A. J. P. Brown, S. Brunke, M. C. Costanzo, D. A. Fitzpatrick, P. W. J. de Groot, D. Harris, L. L. Hoyer, B. Hube, F. M. Klis, C. Kodira, N. Lennard, M. E. Logue, R. Martin, A. M. Neiman, E. Nikolaou, M. A. Quail, J. Quinn, M. C. Santos, F. F. Schmitzberger, G. Sherlock, P. Shah, K. A. T. Silverstein, M. S. Skrzypek, D. Soll, R. Staggs, I. Stansfield, M. P. H. Stumpf, P. E. Sudbery, T. Srikantha, Q. Zeng, J. Berman, M. Berriman, J. Heitman, N. A. R. Gow, M. C. Lorenz, B. W. Birren, M. Kellis, and C. A. Cuomo, "Evolution of pathogenicity and sexual reproduction in eight Candida genomes," *Nature*, vol. 459, no. 7247, pp. 657–662, 2009.

[33] W. M. Fitch, "Homology: a personal view on some of the problems," *Trends in Genetics*, vol. 16, no. 5, pp. 227–231, 2000.

[34] D. Moreira and H. Philippe, "Molecular phylogeny: pitfalls and progress," *International Microbiology*, vol. 3, no. 1, pp. 9–16, 2000.

[35] The Quest for Orthologs Consortium, A. W. Sousa da Silva, B. Boeckmann, C. Dessimoz, E. L. Sonnhammer, M. Robinson-Rechavi, M. Martin, P. D. Thomas, and T. Gabald ón, "Big data and other challenges in the quest for orthologs," *Bioinformatics*, vol. 30, no. 21, pp. 2993–2998, 2014.

[36] M. Lafond, M. Meghdari Miardan, and D. Sankoff, "Accurate prediction of orthologs in the presence of divergence after duplication," *Bioinformatics*, vol. 34, no. 13, pp. i366–i375, 2018.

[37] A. M. Altenhoff, N. M. Glover, and C. Dessimoz, "Inferring Orthology and Paralogy," in *Evolutionary Genomics. Methods in Molecular Biology*, M. Anisimova, Ed. New York, NY, USA: Humana, 2019, vol. 1910, pp. 149–175.

[38] L. Liu and L. Yu, "Estimating Species Trees from Unrooted Gene Trees," *Systematic Biology*, vol. 60, no. 5, pp. 661–667, 2011.

[39] P. Vachaspati and T. Warnow, "ASTRID: Accurate Species Trees from Internode Distances," *BMC Genomics*, vol. 16, no. Suppl 10, p. S3, 2015.

[40] G. Dasarathy, R. Nowak, and S. Roch, “Data Requirement for Phylogenetic Inference from Multiple Loci: A New Distance Method,” *IEEE/ACM Transactions in Computational Biology and Bioinformatics*, vol. 12, no. 2, pp. 422–432, 2015.

[41] E. S. Allman, C. Long, and J. A. Rhodes, “Species Tree Inference from Genomic Sequences Using the Log-Det Distance,” *SIAM Journal on Applied Algebra and Geometry*, vol. 3, no. 1, pp. 107–127, 2019.

[42] J. Heled and A. J. Drummond, “Bayesian Inference of Species Trees from Multilocus Data,” *Molecular Biology and Evolution*, vol. 27, no. 3, pp. 570–580, 2010.

[43] H. A. Ogilvie, R. R. Bouckaert, and A. J. Drummond, “StarBEAST2 Brings Faster Species Tree Inference and Accurate Estimates of Substitution Rates,” *Molecular Biology and Evolution*, vol. 34, no. 8, pp. 2101–2114, 2017.

[44] L. Liu, L. Yu, and S. V. Edwards, “A maximum pseudo-likelihood approach for estimating species trees under the coalescent model,” *BMC Evolutionary Biology*, vol. 10, p. 302, 2010.

[45] A. Stamatakis, “RAxML Version 8: A tool for Phylogenetic Analysis and Post-Analysis of Large Phylogenies,” *Bioinformatics*, vol. 30, no. 9, pp. 1312–1313, 2014.

[46] R. Chaudhary, J. G. Burleigh, and D. Fernández-Baca, “Inferring Species Trees from Incongruent Multi-Copy Gene Trees Using the Robinson-Foulds Distance,” *Algorithms for Molecular Biology*, vol. 8, p. 28, 2013.

[47] R. Chaudhary, D. Fernández-Baca, and J. G. Burleigh, “MulRF: a software package for phylogenetic analysis using multi-copy gene trees,” *Bioinformatics*, vol. 31, no. 3, pp. 432–433, 2014.

[48] M. T. Hallett and J. Lagergren, “New Algorithms for the Duplication-loss Model,” in *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB)*, 2000, pp. 138–146.

[49] S. Mirarab, R. Reaz, M. S. Bayzid, T. Zimmermann, M. S. Swenson, and T. Warnow, “ASTRAL: genome-scale coalescent-based species tree estimation,” *Bioinformatics*, vol. 30, no. 17, pp. i541–i548, 2014.

[50] M.-Y. Chen, D. Liang, and P. Zhang, “Selecting Question-Specific Genes to Reduce Incongruence in Phylogenomics: A Case Study of Jawed Vertebrate Backbone Phylogeny,” *Systematic Biology*, vol. 64, no. 6, pp. 1104–1120, 2015.

[51] P. A. Hosner, B. C. Faircloth, T. C. Glenn, E. L. Braun, and R. T. Kimball, “Avoiding Missing Data Biases in Phylogenomic Inference: An Empirical Study in the Landfowl (Aves: Galliformes),” *Molecular Biology and Evolution*, vol. 33, no. 4, pp. 1110–1125, 2016.

[52] K. A. Meiklejohn, E. L. Braun, R. T. Kimball, B. C. Faircloth, and T. C. Glenn, "Analysis of a Rapid Evolutionary Radiation Using Ultraconserved Elements: Evidence for a Bias in Some Multispecies Coalescent Methods," *Systematic Biology*, vol. 65, no. 4, pp. 612–627, 2016.

[53] S. J. Longo, B. C. Faircloth, A. Meyer, M. W. Westneat, M. E. Alfaro, and P. C. Wainwright, "Phylogenomic analysis of a rapid radiation of misfit fishes (Syngnathiformes) using ultraconserved elements," *Molecular Phylogenetics and Evolution*, vol. 113, pp. 33–48, 2017.

[54] M. P. K. Blom, J. G. Bragg, S. Potter, and C. Moritz, "Accounting for Uncertainty in Gene Tree Estimation: Summary-Coalescent Species Tree Inference in a Challenging Radiation of Australian Lizards," *Systematic Biology*, vol. 66, no. 3, pp. 352–366, 2017.

[55] O. R. Bininda-Emonds, Ed., *Phylogenetic Supertrees: Combining information to reveal the Tree of Life*, ser. Computational Biology. Netherlands: Springer, 2004, vol. 4.

[56] T. Warnow, "Supertree Construction: Opportunities and Challenges," *arXiv*, p. 1805.03530, 2018, available at https://arxiv.org/abs/1805.03530.

[57] B. Legried, E. K. Molloy, T. Warnow, and S. Roch, "Polynomial-Time Statistical Estimation of Species Trees under Gene Duplication and Loss," in *Research in Computational Molecular Biology. RECOMB 2020. Lecture Notes in Computer Science*, R. Schwartz, Ed. Cham, Switzerland: Springer, 2020, vol. 12074, pp. 120–135.

[58] M. Rabiee, E. Sayyari, and S. Mirarab, "Multi-allele species reconstruction using ASTRAL," *Molecular Phylogenetics and Evolution*, vol. 130, pp. 286–296, 2019.

[59] P. Vachaspati and T. Warnow, "FastRFS: fast and accurate Robinson-Foulds Supertrees using constrained exact optimization," *Bioinformatics*, vol. 33, no. 5, pp. 631–639, 2016.

[60] M. D. Rasmussen and M. Kellis, "Unified modeling of gene duplication, loss, and coalescence using a locus tree," *Genome Research*, vol. 22, no. 4, pp. 755–765, 2012.

[61] A. Wehe, M. S. Bansal, J. G. Burleigh, and O. Eulenstein, "DupTree: a program for large-scale phylogenetic analyses using gene tree parsimony," *Bioinformatics*, vol. 24, no. 13, pp. 1540–1541, 2008.

[62] G. Estabrook, C. Johnson, and F. McMorris, "An idealized concept of the true cladistic character," *Mathematical Biosciences*, vol. 23, no. 3–4, pp. 263–272, 1975.

[63] D. F. Robinson and L. R. Foulds, "Comparison of Phylogenetic Trees," *Mathematical Biosciences*, vol. 53, no. 1-2, pp. 131–147, 1981.

[64] Y. Lin, V. Rajan, and B. M. E. Moret, "A Metric for Phylogenetic Trees Based on Matching," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1014–1022, 2012.

[65] M. K. Kuhner and J. Yamato, “Practical Performance of Tree Comparison Metrics,” *Systematic Biology*, vol. 64, no. 2, pp. 205–214, 2014.

[66] D. F. Robinson, “Comparison of labeled trees with valency three,” *Journal of Combinatorial Theory, Series B*, vol. 11, no. 2, pp. 105–119, 1971.

[67] G. W. Moore, M. Goodman, and J. Barnabas, “An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets,” *Journal of Theoretical Biology*, vol. 38, pp. 423–457, 1973.

[68] C. E. Hinchliff, S. A. Smith, J. F. Allman, J. G. Burleigh, R. Chaudhary, L. M. Coghill, K. A. Crandall, J. Deng, B. T. Drew, R. Gazis, K. Gude, D. S. Hibbett, L. A. Katz, H. D. Laughinghouse, E. J. McTavish, P. E. Midford, C. L. Owen, R. H. Ree, J. A. Rees, D. E. Soltis, T. Williams, and K. A. Cranston, “Synthesis of phylogeny and taxonomy into a comprehensive tree of life,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 112, no. 41, pp. 12 764–12 769, 2015.

[69] R. B. D. and H. M. T., “A supertree pipeline for summarizing phylogenetic and taxonomic information for millions of species,” *PeerJ*, vol. 5, p. e3058, 2017.

[70] H. L. Bodlaender, M. R. Fellows, and T. J. Warnow, “Two strikes against perfect phylogeny,” in *Automata, Languages and Programming. ICALP 1992. Lecture Notes in Computer Science*, W. Kuich, Ed. Berlin, Heidelberg: Springer, 1992, vol. 623, pp. 273–283.

[71] M. Steel, “The complexity of reconstructing trees from qualitative characters and subtrees,” *Journal of Classification*, vol. 9, no. 1, pp. 91–116, 1992.

[72] M. A. Ragan, “Phylogenetic inference based on matrix representation of trees,” *Molecular Phylogenetics and Evolution*, vol. 1, no. 1, pp. 53–58, 1992.

[73] T. Jiang, P. Kearney, and M. Li, “A Polynomial Time Approximation Scheme for Inferring Evolutionary Trees from Quartet Topologies and Its Application,” *SIAM Journal on Computing*, vol. 30, no. 6, pp. 1942–1961, 2001.

[74] M. Lafond and C. Scornavacca, “On the Weighted Quartet Consensus problem,” *Theoretical Computer Science*, vol. 769, pp. 1–17, 2019.

[75] K. Strimmer and A. von Haeseler, “Quartet Puzzling: A Quartet Maximum-Likelihood Method for Reconstructing Tree Topologies,” *Molecular Biology and Evolution*, vol. 13, no. 7, pp. 964–964, 1996.

[76] S. Snir and S. Rao, “Quartet MaxCut: A fast algorithm for amalgamating quartet trees,” *Molecular Phylogenetics and Evolution*, vol. 62, no. 1, pp. 1–8, 2012.

[77] R. Reaz, M. S. Bayzid, and M. S. Rahman, “Accurate Phylogenetic Tree Reconstruction from Quartets: A Heuristic Approach,” *PLoS ONE*, vol. 9, no. 8, p. e104008, 2014.

[78] D. Bryant and M. Steel, “Constructing Optimal Trees from Quartets,” *Journal of Algorithms*, vol. 38, no. 1, pp. 237–259, 2001.

[79] M. S. Bansal, J. G. Burleigh, O. Eulenstein, and D. Fernández-Baca, “Robinson-Foulds Supertrees,” *Algorithms for Molecular Biology*, vol. 5, p. 18, 2010.

[80] M. Brinkmeyer, T. Griebel, and S. Böcker, “Polynomial Supertree Methods Revisited,” *Advances in Bioinformatics*, p. 524182, 2011.

[81] M. S. Springer and J. Gatesy, “The gene tree delusion,” *Molecular Phylogenetics and Evolution*, vol. 94, Part A, pp. 1–33, 2016.

[82] G. Bejerano, M. Pheasant, I. Makunin, S. Stephen, W. J. Kent, J. S. Mattick, and D. Haussler, “Ultraconserved Elements in the Human Genome,” *Science*, vol. 304, no. 5675, pp. 1321–1325, 2004.

[83] B. C. Faircloth, “Identifying conserved genomic elements and designing universal bait sets to enrich them,” *Methods in Ecology and Evolution*, vol. 8, no. 9, pp. 1103–1112, 2017.

[84] D. Posada, K. A. Crandall, and E. C. Holmes, “Recombination in evolutionary genomics,” *Annual Review of Genetics*, vol. 36, no. 1, pp. 75–97, 2002.

[85] R. C. Griffiths and P. Marjoram, “Ancestral Inference from Samples of DNA Sequences with Recombination,” *Journal of Computational Biology*, vol. 3, no. 4, pp. 479–502, 1996.

[86] D. Posada and K. A. Crandall, “Intraspecific gene genealogies: trees grafting into networks,” *Trends in Ecology and Evolution*, vol. 16, no. 1, pp. 37–45, 2001.

[87] J. Gogarten and J. Townsend, “Horizontal gene transfer, genome innovation and evolution,” *Nature Reviews Microbiology*, vol. 3, pp. 679–687, 2005.

[88] L. Boto, “Horizontal gene transfer in evolution: facts and challenges,” *Proceedings of the Royal Society B*, vol. 277, pp. 819–827, 2009.

[89] M. Pérez-Losada, M. Arenas, J. C. Galán, F. Palero, and F. González-Candelas, “Recombination in viruses: Mechanisms, methods of study, and evolutionary consequences,” *Infection, Genetics and Evolution*, vol. 30, pp. 296–307, 2015.

[90] J. J. Wiens, “Species Delimitation: New Approaches for Discovering Diversity,” *Systematic Biology*, vol. 56, no. 6, pp. 875–878, 2007.

[91] B. C. O’Meara, “New Heuristic Methods for Joint Species Delimitation and Species Tree Inference,” *Systematic Biology*, vol. 59, no. 1, pp. 59–73, 11 2009.

[92] C. Solís-Lemus, L. Knowles, and C. Ané, “Bayesian species delimitation combining multiple genes and traits in a unified framework,” *Evolution*, vol. 69, pp. 492–507, 2015.

[93] M. Rabiee and S. Mirarab, "SODA: Multi-Locus Species Delimitation Using Quartet Frequencies," *bioRxiv*, p. 869396, 2019, available at https://dx.doi.org/10.1101/869396.

[94] J. C. Avise and K. Wollenberg, "Phylogenetics and the origin of species," *Proceedings of the National Academy of Sciences of the United of America*, vol. 94, no. 15, pp. 7748–7755, 1997.

[95] L. Nakhleh, *Evolutionary Phylogenetic Networks: Models and Issues.* Boston, MA: Springer, 2011, pp. 125–158.

[96] D. H. Huson, R. Rupp, and C. Scornavacca, *Phylogenetic Networks: Concepts, Algorithms and Applications.* Cambridge: Cambridge University Press, 2010.

[97] J. F. C. Kingman, "On the Genealogy of Large Populations," *Journal of Applied Probability*, vol. 19, no. 1982, pp. 27–43, 1982.

[98] J. Wakeley, *Coalescent Theory: An Introduction.* Greenwood Village, CO: Roberts & Company Publishers, 2009.

[99] S. Tavaré, "Line-of-descent and genealogical processes, and their applications in population genetics models," *Theoretical Population Biology*, vol. 26, pp. 119–164, 1984.

[100] E. S. Allman, J. H. Degnan, and J. A. Rhodes, "Identifying the rooted species tree from the distribution of unrooted gene trees under the coalescent," *Journal of Mathematical Biology*, vol. 62, no. 6, pp. 833–862, 2011.

[101] J. H. Degnan and N. A. Rosenberg, "Discordance of Species Trees with Their Most Likely Gene Trees," *PLoS Genetics*, vol. 2, pp. 762–768, 2006.

[102] J. H. Degnan, "Anomalous Unrooted Gene Trees," *Systematic Biology*, vol. 62, no. 4, pp. 574–590, 2013.

[103] D. G. Kendall, "On the Generalized "Birth-and-Death" Process," *The Annals of Mathematical Statistics*, vol. 19, no. 1, pp. 1–15, 1948.

[104] S. Nee, R. M. May, and P. H. Harvey, "The reconstructed evolutionary process," *Philosophical Transactions of the Royal Society of London B*, vol. 344, no. 1309, pp. 305–311, 1994.

[105] P. Tataru, M. Simonsen, T. Bataillon, and A. Hobolth, "Statistical Inference in the Wright–Fisher Model Using Allele Frequency Data," *Systematic Biology*, vol. 66, no. 1, pp. e30–e46, 2016.

[106] A. Rambaut and N. C. Grass, "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees," *Bioinformatics*, vol. 13, no. 3, pp. 235–238, 1997.

[107] C. Tuffley and M. Steel, “Links between maximum likelihood and maximum parsimony under a simple model of site substitution,” *Bulletin of Mathematical Biology*, vol. 59, no. 3, pp. 581–607, 1997.

[108] Z. Yang, *Computational Molecular Evolution.* New York, NY: Oxford University Press, 2006.

[109] M. Steel, “Recovering a tree from the leaf colourations it generates under a Markov model,” *Applied Mathematics Letters*, vol. 7, no. 2, pp. 19–23, 1994.

[110] Z. Yang, “,” *Journal of Molecular Evolution*, vol. 39, no. 3, pp. 306–314, Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods.

[111] Z. Yang, “Among-site rate variation and its impact on phylogenetic analyses,” *Trends in Ecology and Evolution*, vol. 11, no. 9, pp. 367–372, 1996.

[112] S. Ho, “The Molecular Clock and Estimating Species Divergence,” *Nature Education*, vol. 1, no. 1, p. 168, 2008.

[113] A. J. Drummond, S. Y. W. Ho, M. J. Phillips, and A. Rambaut, “Relaxed Phylogenetics and Dating with Confidence,” *PLOS Biology*, vol. 4, no. 5, p. e88, 2006.

[114] L. Bromham and D. Penny, “The modern molecular clock,” *Nature Review Genetics*, vol. 4, p. 216–224, 2003.

[115] E. K. Molloy and T. Warnow, “Large-scale Species Tree Estimation,” in , L. L. Knowles and L. Kubatko, Eds., 2020, under revision.

[116] J. Truszkowski and N. Goldman, “Maximum Likelihood Phylogenetic Inference is Consistent on Multiple Sequence Alignments, with or without Gaps,” *Systematic Biology*, vol. 65, no. 2, pp. 328–333, 2016.

[117] S. Roch, “A short proof that phylogenetic tree reconstruction by maximum likelihood is hard,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 1, pp. 92–94, 2006.

[118] L.-T. Nguyen, H. Schmidt, A. von Haeseler, and B. Minh, “IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies,” *Molecular Biology and Evolution*, vol. 32, no. 1, pp. 268–274, 2015.

[119] M. N. Price, P. S. Dehal, and A. P. Arkin, “FastTree 2 - Approximately Maximum-Likelihood Trees for Large Alignments,” *PLoS ONE*, vol. 5, no. 3, pp. 1–10, 2010.

[120] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel, “New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0,” *Systematic Biology*, vol. 59, no. 3, pp. 307–321, 2010.

[121] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406–425, 1987.

[122] K. Atteson, "The Performance of Neighbor-Joining Methods of Phylogenetic Reconstruction," *Algorithmica*, vol. 25, no. 2-3, pp. 251–278, 1999.

[123] M. A. Steel, "Recovering a tree from the leaf colourations it generates under a Markov model," *Applied Mathematics Letters*, vol. 7, no. 2, pp. 19–24, 1994.

[124] J. Felsenstein, "Cases in which parsimony or compatibility methods will be positively misleading," *Systematic Zoology*, vol. 1978, no. 7, pp. 401–410, 1978.

[125] S. Roch, M. Nute, and T. Warnow, "Long-Branch Attraction in Species Tree Estimation: Inconsistency of Partitioned Likelihood and Topology-Based Summary Methods," *Systematic Biology*, vol. 68, no. 2, pp. 281–297, 2018.

[126] M. Hendy and D. Penny, "A framework for the quantitative study of evolutionary trees," *Systematic Biology*, vol. 38, no. 4, p. 297–309, 1989.

[127] T. A. Heath, S. M. Hedtke, and D. M. Hillis, "Taxon sampling and the accuracy of phylogenetic analyses," *Journal of Systematics and Evolution*, vol. 46, no. 3, pp. 239–257, 2008.

[128] J. Felsenstein, "Confidence Limits on Phylogenies: An Approach Using the Bootstrap," *Evolution*, vol. 39, no. 4, pp. 783–791, 1985.

[129] B. Efron, E. Halloran, and S. Holmes, "Bootstrap confidence levels for phylogenetic trees," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 23, pp. 13 429–13 429, 1996.

[130] S. Holmes, "Bootstrapping Phylogenetic Trees: Theory and Methods," *Statistical Science*, vol. 18, no. 2, pp. 241–255, 2003.

[131] D. L. Swofford, G. J. Olson, P. J. Waddell, and D. M. Hillis, *Phylogenetic Inference*, 2nd ed. Sunderland, MA: Sinauer Associates, 1996.

[132] E. K. Molloy and T. Warnow, "To include or not to include: The impact of gene filtering on species tree estimation methods," *Systematic Biology*, vol. 67, no. 2, pp. 285–303, 2018.

[133] L. S. Kubatko and J. H. Degnan, "Inconsistency of Phylogenetic Estimates from Concatenated Data under Coalescence," *Systematic Biology*, vol. 56, no. 1, pp. 17–24, 2007.

[134] S. Roch and M. Steel, "Likelihood-based tree reconstruction on a concatenation of aligned sequence data sets can be statistically inconsistent," *Theoretical Population Biology*, vol. 100, pp. 56–62, 2015.

[135] M. Wascher and L. Kubatko, "Consistency of SVDQuartets and Maximum Likelihood for Coalescent-based Species Tree Estimation," *bioRxiv*, 2019, available at https://dx.doi.org/10.1101/523050.

[136] A. M. Kozlov, A. J. Aberer, and A. Stamatakis, "ExaML version 3: a tool for phylogenomic analyses on supercomputers," *Bioinformatics*, vol. 31, no. 15, pp. 2577–2579, 2015.

[137] J. Demmel and K. Yelick, *Communication Avoiding (CA) and Other Innovative Algorithms.* Microsoft Corporation, 2013, pp. 243–250.

[138] T. Hoefler, A. Lumsdaine, and W. Rehm, "Implementation and performance analysis of non-blocking collective operations for MPI," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, ser. SC '07. ACM, 2007, pp. 1–10.

[139] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," in *Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPOPP '93. New York, NY, USA: ACM, 1993, pp. 1–12.

[140] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: Incorporating Long Messages into the LogP Model — One Step Closer Towards a Realistic Model for Parallel Computation," University of California at Santa Barbara, Santa Barbara, CA, USA, Tech. Rep., 1995.

[141] W. Gropp, L. N. Olson, and P. Samfass, "Modeling MPI Communication Performance on SMP Nodes: Is It Time to Retire the Ping Pong Test," in *Proceedings of the 23rd European MPI Users' Group Meeting*, ser. EuroMPI 2016. New York, NY, USA: ACM, 2016, pp. 41–50.

[142] S. V. Edwards, L. Liu, and D. K. Pearl, "High-resolution species trees without concatenation," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 14, pp. 5936–5941, 2007.

[143] L. Liu, "BEST: Bayesian estimation of species trees under the coalescent model," *Bioinformatics*, vol. 24, no. 21, pp. 2542–2543, 2008.

[144] A. D. Leaché and B. Rannala, "The Accuracy of Species Tree Estimation under Simulation: A Comparison of Methods," *Systematic Biology*, vol. 60, no. 2, pp. 126–137, 2010.

[145] M. S. Bayzid and T. Warnow, "Naive binning improves phylogenomic analyses," *Bioinformatics*, vol. 29, no. 18, pp. 2277–2284, 2013.

[146] J. E. McCormack, H. Huang, and L. L. Knowles, "Maximum Likelihood Estimates of Species Trees: How Accuracy of Phylogenetic Inference Depends upon the Divergence History and Sampling Design," *Systematic Biology*, vol. 58, no. 5, pp. 501–508, 2009.

[147] T. Zimmermann, S. Mirarab, and T. Warnow, “BBCA: Improving the scalability of *BEAST using random binning,” *BMC Genomics*, vol. 15, no. Suppl 6, p. S11, 2014.

[148] S. D. Leavitt, F. Grewe, T. Widhelm, L. Muggia, B. Wray, and H. T. Lumbsch, “Resolving evolutionary relationships in lichen-forming fungi using diverse phylogenomic datasets and analytical approaches,” *Scientific Reports*, vol. 6, p. 22262, 2016.

[149] L. Liu, L. Yu, D. K. Pearl, and S. V. Edwards, “Estimating Species Phylogenies Using Coalescence Times among Sequences,” *Systematic Biology*, vol. 58, no. 5, pp. 468–477, 2009.

[150] L. S. Kubatko, B. C. Carstens, and L. L. Knowles, “STEM: species tree estimation using maximum likelihood for gene trees under coalescence,” *Bioinformatics*, vol. 25, no. 7, pp. 971–973, 2009.

[151] E. M. Jewett and N. A. Rosenberg, “iGLASS: An Improvement to the GLASS Method for Estimating Species Trees from Gene Trees,” *Journal of Computational Biology*, vol. 19, no. 3, pp. 293–315, 2012.

[152] S. Mirarab and T. Warnow, “ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes,” *Bioinformatics*, vol. 31, no. 12, pp. i44–i52, 2015.

[153] E. S. Allman, J. H. Degnan, and J. A. Rhodes, “Species Tree Inference from Gene Splits by Unrooted STAR Methods,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 1, pp. 337–342, 2018.

[154] V. Lefort, R. Desper, and O. Gascuel, “FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program,” *Molecular Biology and Evolution*, vol. 32, no. 10, pp. 2798–2800, 2015.

[155] R. Desper and O. Gascuel, “Fast and accurate phylogeny reconstruction algorithm based on the minimum-evolution principle,” *Journal of Computational Biology*, vol. 9, no. 2, pp. 687–705, 2004.

[156] L.-S. Wang, T. Warnow, B. M. E. Moret, R. K. Jansen, and L. A. Raubeson, “Distance-based genome rearrangement phylogeny,” *Journal of Molecular Evolution*, vol. 63, no. 4, pp. 473–483, 2006.

[157] A. Criscuolo and O. Gascuel, “Fast NJ-like algorithms to deal with incomplete distance matrices,” *BMC Bioinformatics*, vol. 9, p. 166, 2008.

[158] C. Zhang, M. Rabiee, E. Sayyari, and S. Mirarab, “ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees,” *BMC Bioinformatics*, vol. 19, no. Suppl 6, p. 153, 2018.

[159] D. Kane and T. Tao, “A bound on partitioning clusters,” *arXiv*, 2017, available at https://arxiv.org/abs/1702.00912.

[160] S. Patel, R. T. Kimball, and E. L. Braun, “Error in Phylogenetic Estimation for Bushes in the Tree of Life,” *Journal of Phylogenetics and Evolutionary Biology*, vol. 1, p. 110, 2013.

[161] S. Mirarab, M. S. Bayzid, B. Boussau, and T. Warnow, “Statistical binning enables an accurate coalescent-based estimation of the avian tree,” *Science*, vol. 346, no. 6215, p. 1250463, 2014.

[162] M. S. Bayzid, S. Mirarab, B. Boussau, and T. Warnow, “Weighted Statistical Binning: Enabling Statistically Consistent Genome-Scale Phylogenetic Analyses,” *PLoS ONE*, vol. 10, no. 6, p. e0129183, 2015.

[163] J. Chou, A. Gupta, S. Yaduvanshi, R. Davidson, M. Nute, S. Mirarab, and T. Warnow, “A comparative study of SVDquartets and other coalescent-based species tree estimation methods,” *BMC Genomics*, vol. 16, no. Suppl 10, p. S2, 2015.

[164] S. Mirarab, M. S. Bayzid, and T. Warnow, “Evaluating summary methods for multilocus species tree estimation in the presence of incomplete lineage sorting,” *Systematic Biology*, vol. 65, no. 3, pp. 366–380, 2016.

[165] S. Roch and T. Warnow, “On the Robustness to Gene Tree Estimation Error (or lack thereof) of Coalescent-Based Species Tree Methods,” *Systematic Biology*, vol. 64, no. 4, pp. 663–676, 2015.

[166] H. Huang, Q. He, L. S. Kubatko, and L. L. Knowles, “Sources of Error Inherent in Species-Tree Estimation: Impact of Mutational and Coalescent Effects on Accuracy and Implications for Choosing among Different Methods,” *Systematic Biology*, vol. 59, no. 5, pp. 573–583, 2010.

[167] M. DeGiorgio and J. H. Degnan, “Robustness to Divergence Time Underestimation When Inferring Species Trees from Estimated Gene Trees,” *Systematic Biology*, vol. 63, no. 1, pp. 66–82, 2014.

[168] H. C. Lanier and L. L. Knowles, “Applying species-tree analyses to deep phylogenetic histories: Challenges and potential suggested from a survey of empirical phylogenetic studies,” *Molecular Phylogenetics and Evolution*, vol. 83, pp. 191–199, 2015.

[169] Z. Xi, L. Liu, and C. C. Davis, “Genes with minimal phylogenetic information are problematic for coalescent analyses when gene tree estimation is biased,” *Molecular Phylogenetics and Evolution*, vol. 92, pp. 63–71, 2015.

[170] A. C. Driskell, C. Ané, J. G. Burleigh, M. M. McMahon, B. C. O’Meara, and M. J. Sanderson, “Prospects for Building the Tree of Life from Large Sequence Databases,” *Science*, vol. 306, no. 5699, pp. 1172–1174, 2004.

[171] J. W. Streicher, J. A. Schulte, and J. J. Wiens, “How Should Genes and Taxa be Sampled for Phylogenomic Analyses with Missing Data? An Empirical Study in Iguanian Lizards,” *Systematic Biology*, vol. 65, no. 1, pp. 128–145, 2016.

[172] Z. Xi, L. Liu, and C. C. Davis, "The Impact of Missing Data on Species Tree Estimation," *Molecular Biology and Evolution*, vol. 33, no. 3, pp. 838–860, 2016.

[173] M. Nute, J. Chou, E. K. Molloy, and T. Warnow, "The performance of coalescent-based species tree estimation methods under models of missing data," *BMC Genomics*, vol. 19(Suppl 5), p. 286, 2018.

[174] J. A. Rhodes, M. G. Nute, and T. Warnow, "NJst and ASTRID are not statistically consistent under a random model of missing data," *arXiv*, p. 2001.07844, 2020, available at https://arxiv.org/abs/2001.07844.

[175] R. Hovmöller, L. Lacey, and L. S. Kubatko, "Effects of missing data on species tree estimation under the coalescent," *Molecular Phylogenetics and Evolution*, vol. 69, no. 3, pp. 1057–1062, 2013.

[176] D. Bryant, R. Bouckaert, J. Felsenstein, N. A. Rosenberg, and A. RoyChoudhury, "Inferring Species Trees Directly from Biallelic Genetic Markers: Bypassing Gene Trees in a Full Coalescent Analysis," *Molecular Biology and Evolution*, vol. 29, no. 8, pp. 1917–1932, 2012.

[177] J. Chifman and L. Kubatko, "Quartet Inference from SNP Data Under the Coalescent Model," *Bioinformatics*, vol. 30, no. 23, pp. 3317–3324, 2014.

[178] J. Chifman and L. Kubatko, "Identifiability of the unrooted species tree topology under the coalescent model with time-reversible substitution processes, site-specific rate variation, and invariable sites," *Journal of Theoretical Biology*, vol. 374, pp. 35–47, 2015.

[179] D. L. Swofford, "PAUP* (*Phylogenetic Analysis Using PAUP)." [Online]. Available: http://phylosolutions.com/paup-test/

[180] M. S. Swenson, R. Suri, C. R. Linder, and T. Warnow, "An experimental study of Quartets MaxCut and other supertree methods," *Algorithms for Molecular Biology*, vol. 6, p. 7, 2011.

[181] J. Gatesy and M. S. Springer, "Phylogenetic analysis at deep timescales: Unreliable gene trees, bypassed hidden support, and the coalescence/concatalescence conundrum," *Molecular Biology and Evolution*, vol. 80, pp. 231–266, 2014.

[182] J. J. Wiens and M. C. Morrill, "Missing Data in Phylogenetic Analysis: Reconciling Results from Simulations and Empirical Data," *Systematic Biology*, vol. 60, no. 5, pp. 719–731, 2011.

[183] S. Cho, A. Zwick, J. C. Regier, C. Mitter, M. P. Cummings, J. Yao, Z. Du, H. Zhao, A. Y. Kawahara, S. Weller, D. R. Davis, J. Baixeras, J. W. Brown, and C. Parr, "Can Deliberately Incomplete Gene Sample Augmentation Improve a Phylogeny Estimate for the Advanced Moths and Butterflies (Hexapoda: Lepidoptera)?" *Systematic Biology*, vol. 60, no. 6, pp. 782–796, 2011.

[184] L. Salichos and A. Rokas, "Inferring ancient divergences requires genes with strong phylogenetic signals," *Nature*, vol. 497, no. 7449, pp. 327–331, 2013.

[185] L. Salichos, A. Stamatakis, and A. Rokas, "Novel Information Theory-Based Measures for Quantifying Incongruence among Phylogenetic Trees," *Molecular Biology and Evolution*, vol. 31, no. 5, pp. 1261–1271, 2014.

[186] R. Betancur-R, G. J. Naylor, and G. Ortí, "Conserved Genes, Sampling Error, and Phylogenomic Inference," *Systematic Biology*, vol. 63, no. 2, pp. 257–262, 2014.

[187] W. Jiang, S.-Y. Chen, H. Wang, D.-Z. Li, and J. J. Wiens, "Should genes with missing data be excluded from phylogenetic analyses?" *Molecular Phylogenetics and Evolution*, vol. 80, no. Supplement C, pp. 308–318, 2014.

[188] A. Dornburg, J. P. Townsend, M. Friedman, and T. J. Near, "Phylogenetic informativeness reconciles ray-finned fish molecular divergence times," *BMC Evolutionary Biology*, vol. 14, p. 169, 2014.

[189] J. W. Streicher and J. J. Wiens, "Phylogenomic analyses reveal novel relationships among snake families," *Molecular Phylogenetics and Evolution*, vol. 100, pp. 160–169, 2016.

[190] A. Dornburg, J. P. Townsend, W. Brooks, E. Spriggs, R. I. Eytan, J. A. Moore, P. C. Wainwright, A. Lemmon, E. M. Lemmon, and T. J. Near, "New insights on the sister lineage of percomorph fishes with an anchored hybrid enrichment dataset," *Molecular Phylogenetics and Evolution*, vol. 110, pp. 27–38, 2017.

[191] L. Liu, Z. Xi, S. Wu, C. C. Davis, and S. V. Edwards, "Estimating phylogenetic trees from genome-scale data," *Annals of the New York Academy of Sciences*, vol. 1360, no. 1, pp. 36–53, 2015.

[192] H. Huang and L. L. Knowles, "Unforeseen Consequences of Excluding Missing Data from Next-Generation Sequences: Simulation Study of RAD Sequences," *Systematic Biology*, vol. 65, no. 3, pp. 357–365, 2016.

[193] M. P. Simmons, D. B. Sloan, and J. Gatesy, "The effects of subsampling gene trees on coalescent methods applied to ancient divergences," *Molecular Phylogenetics and Evolution*, vol. 97, pp. 76–89, 2016.

[194] H. C. Lanier, H. Huang, and L. L. Knowles, "How low can you go? The effects of mutation rate on the accuracy of species-tree estimation," *Molecular Phylogenetics and Evolution*, vol. 70, pp. 112–119, 2014.

[195] D. Mallo, L. D. O. Martins, and D. Posada, "SimPhy: phylogenomic simulation of gene, locus, and species trees," *Systematic Biology*, vol. 65, no. 2, pp. 334–344, 2016.

[196] M. S. Bayzid, T. Hunt, and T. Warnow, "Disk covering methods improve phylogenomic analyses," *BMC Genomics*, vol. 15, no. Suppl 6, p. S7, 2014.

[197] W. Fletcher and Z. Yang, "INDELible: A Flexible Simulator of Biological Sequence Evolution," *Molecular Biology and Evolution*, vol. 26, no. 8, pp. 1879–1888, 2009.

[198] A. Hobolth, J. Y. Dutheil, J. Hawks, and T. Mailund, "Incomplete lineage sorting patterns among human, chimpanzee, and orangutan suggest recent orangutan speciation and widespread selection," *Genome Research*, vol. 21, no. 3, pp. 349–356, 2011.

[199] E. Sayyari and S. Mirarab, "Fast Coalescent-Based Computation of Local Branch Support from Quartet Frequencies," *Molecular Biology and Evolution*, vol. 33, no. 7, pp. 1654–1668, 2016.

[200] R. Davidson, P. Vachaspati, S. Mirarab, and T. Warnow, "Phylogenomic species tree estimation in the presence of incomplete lineage sorting and horizontal gene transfer," *BMC Genomics*, vol. 16, no. Suppl 10, p. S1, 2015.

[201] L. Liu, S. Wu, and L. Yu, "Coalescent methods for estimating species trees from phylogenomic data," *Journal of Systematics and Evolution*, vol. 53, no. 5, pp. 380–390, 2015.

[202] S. Shekhar, S. Roch, and S. Mirarab, "Species tree estimation using ASTRAL: how many genes are enough?" *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 15, no. 5, pp. 1738–1747, 2018.

[203] F. Ronquist and J. P. Huelsenbeck, "MrBayes 3: Bayesian phylogenetic inference under mixed models," *Bioinformatics*, vol. 19, no. 12, pp. 1572–1574, 2003.

[204] N. A. Baird, P. D. Etter, T. S. Atwood, M. C. Currey, A. L. Shiver, Z. A. Lewis, E. U. Selker, W. A. Cresko, and E. A. Johnson, "Rapid SNP Discovery and Genetic Mapping Using Sequenced RAD Markers," *PLoS ONE*, vol. 3, p. e3376, 2008.

[205] J. P. Townsend and C. Leuenberger, "Taxon Sampling and the Optimal Rates of Evolution for Phylogenetic Inference," *Systematic Biology*, vol. 60, no. 3, pp. 358–365, 2011.

[206] M. Anisimova, M. Gil, J.-F. Dufayard, C. Dessimoz, and O. Gascuel, "Survey of Branch Support Methods Demonstrates Accuracy, Power, and Robustness of Fast Likelihood-based Approximation Schemes," *Systematic Biology*, vol. 60, no. 5, pp. 685–699, 2011.

[207] S. Holmes, "Statistical Approach to Tests Involving Phylogenies," in *Mathematics of Evolution and Phylogeny*, O. Gascuel, Ed. Oxford, UK: Oxford University Press, 2005, pp. 91–120.

[208] E. Susko, "Bootstrap Support Is Not First-Order Correct," *Systematic Biology*, vol. 58, no. 2, pp. 211–223, 2009.

[209] M. Fischer and M. Steel, "Sequence length bounds for resolving a deep phylogenetic divergence," *Journal of Theoretical Biology*, vol. 256, no. 2, pp. 247–252, 2009.

[210] J. P. Townsend, Z. Su, and Y. I. Tekle, "Phylogenetic Signal and Noise: Predicting the Power of a Data Set to Resolve Phylogeny," *Systematic Biology*, vol. 61, no. 5, pp. 835–849, 2012.

[211] E. Susko and A. J. Roger, "The Probability of Correctly Resolving a Split as an Experimental Design Criterion in Phylogenetics," *Systematic Biology*, vol. 61, no. 5, pp. 811–821, 2012.

[212] B. Boussau, G. J. Szöllősi, L. Duret, M. Gouy, E. Tannier, and V. Daubin, "Genome-scale coestimation of species and gene trees," *Genome Research*, vol. 23, no. 2, pp. 323–330, 2013.

[213] S. Roch and S. Snir, "Recovering the Tree-Like Trend of Evolution Despite Extensive Lateral Genetic Transfer: A Probabilistic Analysis," *Journal of Computational Biology*, vol. 20, no. 2, pp. 93–112, 2013.

[214] Y. Yu, J. Dong, K. J. Liu, and L. Nakhleh, "Maximum likelihood inference of reticulate evolutionary histories," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, no. 46, pp. 16 448–16 453, 2014.

[215] C. Solís-Lemus and C. Ané, "Inferring Phylogenetic Networks with Maximum Pseudolikelihood under Incomplete Lineage Sorting," *PLoS Genetics*, vol. 12, no. 3, p. e1005896, 2016.

[216] C. Solís-Lemus, M. Yang, and C. Ané, "Inconsistency of Species Tree Methods under Gene Flow," *Systematic Biology*, vol. 65, no. 5, pp. 843–851, 2016.

[217] D. Wen, Y. Yu, and L. Nakhleh, "Bayesian Inference of Reticulate Phylogenies under the Multispecies Network Coalescent," *PLoS Genetics*, vol. 12, no. 5, p. e1006006, 2016.

[218] J. Zhu, D. Wen, Y. Yu, H. M. Meudt, and L. Nakhleh, "Bayesian inference of phylogenetic networks from bi-allelic genetic markers," *PLOS Computational Biology*, vol. 14, no. 1, pp. 1–32, 2018.

[219] G. Dasarathy, E. Mossel, R. Nowak, and S. Roch, "Coalescent-based species tree estimation: a stochastic Farris transform," *arXiv*, p. 1707.04300, 2017, available at https://arxiv.org/abs/1707.04300.

[220] E. K. Molloy and T. Warnow, "Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge," *Algorithms for Molecular Biology*, vol. 14, p. 14, 2019.

[221] T. Warnow, B. M. E. Moret, and K. St. John, "Absolute convergence: True trees from short sequences," in *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001, pp. 186–195.

[222] D. H. Huson, L. Vawter, and T. Warnow, "Solving Large Scale Phylogenetic Problems Using DCM2," in *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 1999, pp. 118–129.

[223] L. Nakhleh, U. Roshan, K. St. John, J. Sun, and T. Warnow, "Designing fast converging phylogenetic methods," *Bioinformatics*, vol. 17, no. Suppl 1, pp. S190–S198, 2001.

[224] J. Lagergren, "Combining polynomial running time and fast convergence for the disk-covering method," *Journal of Computer and System Science*, vol. 65, no. 3, pp. 481–493, 2002.

[225] M. J. Sanderson, M. M. McMahon, and M. Steel, "Terraces in Phylogenetic Tree Space," *Science*, vol. 333, no. 6041, pp. 448–450, 2011.

[226] M. Swenson, R. Suri, C. R. Linder, and T. Warnow, "SuperFine: fast and accurate supertree estimation," *Systematic Biology*, vol. 61, no. 2, pp. 214–227, 2012.

[227] S. Nelesen, K. Liu, L.-S. Wang, C. R. Linder, and T. Warnow, "DACTAL: divide-and-conquer trees (almost) without alignments," *Bioinformatics*, vol. 28, no. 12, pp. i274–i282, 2012.

[228] O. R. P. Bininda-Emonds and A. Stamatakis, "Taxon sampling versus computational complexity and their impact on obtaining the Tree of Life," April 2020. [Online]. Available: https://cme.h-its.org/exelixis/pubs/TAXON-SAMPLING.pdf

[229] T. J. Warnow, "Tree Compatibility and Inferring Evolutionary History," *Journal of Algorithms*, vol. 16, no. 3, pp. 388–407, 1994.

[230] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman, "Inferring a Tree from Lowest Common Ancestors with an Application to the Optimization of Relational Expressions," *SIAM Journal on Computing*, vol. 10, no. 3, pp. 405–421, 1981.

[231] Y. Deng and D. Fernández-Baca, "Fast Compatibility Testing for Rooted Phylogenetic Trees," *Algorithmica*, vol. 80, pp. 2453–2477, 2018.

[232] W. Day, "Optimal algorithms for comparing trees with labeled leaves," *Journal of Classifaction*, vol. 2, no. 1, pp. 7–28, 1985.

[233] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, "PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences," *Journal of Computational Biology*, vol. 22, no. 5, pp. 377–386, 2015.

[234] D. S. Mitrinović, *Analytic Inequalities.* New York: Springer-Verlag, 1970.

[235] E. K. Molloy and T. Warnow, "NJMerge: A Generic Technique for Scaling Phylogeny Estimation Methods and Its Application to Species Trees," in *Comparative Genomics. RECOMB-CG 2018. Lecture Notes in Computer Science*, M. Blanchette and A. Ouangraoua, Eds. Cham, Switzerland: Springer, 2018, vol. 11183, pp. 260–276.

[236] E. K. Molloy and T. Warnow, "TreeMerge: a new method for improving the scalability of species tree estimation methods," *Bioinformatics*, vol. 35, no. 14, pp. i417–i426, 2019.

[237] D. Bryant and P. Waddell, "Rapid Evaluation of Least-Squares and Minimum-Evolution Criteria on Phylogenetic Trees," *Molecular Biology and Evolution*, vol. 15, no. 10, pp. 1346–1346, 1998.

[238] M. S. Bansal, "Linear-Time Algorithms for Some Phylogenetic Tree Completion Problems Under Robinson-Foulds Distance," in *Comparative Genomics. RECOMB-CG 2018. Lecture Notes in Computer Science*, M. Blanchette and A. Ouangraoua, Eds. Cham, Switzerland: Springer, 2018, vol. 11183, pp. 209–226.

[239] J. B. Kruskal, "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[240] J. Sukumaran and M. T. Holder, "DendroPy: a Python library for phylogenetic computing," *Bioinformatics*, vol. 26, no. 12, pp. 1569–1571, 2010.

[241] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow, "Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees," *Science*, vol. 324, no. 5934, pp. 1561–1564, 2009.

[242] K. Liu, T. J. Warnow, M. T. Holder, S. M. Nelesen, J. Yu, A. P. Stamatakis, and C. R. Linder, "SATé-II: Very Fast and Accurate Simultaneous Estimation of Multiple Sequence Alignments and Phylogenetic Trees," *Systematic Biology*, vol. 61, no. 1, pp. 90–106, 2012.

[243] R. O. Prum, J. S. Berv, A. Dornburg, D. J. Field, J. P. Townsend, E. M. Lemmon, and A. R. Lemmon, "A comprehensive phylogeny of birds (Aves) using targeted next-generation DNA sequencing," *Nature*, vol. 526, pp. 569–573, 2015.

[244] P. Lopez, D. Casane, and H. Philippe, "Heterotachy, an important process of protein evolution," *Molecular Biology and Evolution*, vol. 19, no. 1, pp. 1–7, 2002.

[245] L. S. Jermiin, S. Y. Ho, F. Ababneh, J. Robinson, and A. W. Larkum, "The biasing effect of compositional heterogeneity on phylogenetic estimates may be underestimatedy," *Systematic Biology*, vol. 53, no. 4, pp. 638–643, 2004.

[246] P. Lockhart, P. Novis, B. G. Milligan, J. Riden, A. Rambaut, and T. Larkum, "Heterotachy and tree building: a case study with plastids and eubacteria," *Molecular Biology and Evolution*, vol. 23, no. 1, pp. 40–45, 2005.

[247] Y. Zhou, N. Rodrigue, N. Lartillot, , and H. Philippe, "Evaluation of the models handling heterotachy in phylogenetic inference," *BMC Evolutionary Biology*, vol. 7, no. 1, p. 206, 2007.

[248] S. M. Crotty, B. Q. Minh, N. G. Bean, B. R. Holland, J. Tuke, L. S. Jermiin, and A. V. Haeseler, “GHOST: Recovering Historical Signal from Heterotachously Evolved Sequence Alignments,” *Systematic Biology*, vol. 69, no. 2, pp. 249–264, 2020.

[249] T. Le, A. Sy, E. K. Molloy, Q. Zhang, S. Rao, and T. Warnow, “Using INC Within Divide-and-Conquer Phylogeny Estimation,” in *Algorithms for Computational Biology. AlCoB 2019. Lecture Notes in Computer Science*, I. Holmes, C. Martín-Vide, and M. Vega-Rodríguez, Eds. Cham, Switzerland: Springer, 2018, vol. 11488, pp. 167–178.

[250] Q. R. Zhang, S. Rao, and T. J. Warnow, “Constrained incremental tree building: new absolute fast converging phylogeny estimation methods with improved scalability and accuracy,” *Algorithms for Molecular Biology*, vol. 14, p. 2, 2019.

[251] E. K. Molloy and T. Warnow, “FastMulRFS: fast and accurate species tree estimation under generic gene duplication and loss models,” *Bioinformatics*, 2020, conditionally accepted. Preprint available on at https://doi.org/10.1101/835553.

[252] L. De Oliveira Martins, D. Mallo, and D. Posada, “A Bayesian supertree model for genome-wide species tree reconstruction,” *Systematic Biology*, vol. 65, no. 3, pp. 397–416, 2016.

[253] R. Chaudhary, M. S. Bansal, A. Wehe, D. Fernández-Baca, and O. Eulenstein, “iGTP: a software package for large-scale gene tree parsimony analysis,” *BMC Bioinformatics*, vol. 11, p. 574, 2010.

[254] M. S. Bayzid and T. Warnow, “Gene tree parsimony for incomplete gene trees: addressing true biological loss,” *Algorithms for Molecular Biology*, vol. 13, p. 1, 2018.

[255] M. J. Sanderson and M. M. McMahon, “Inferring angiosperm phylogeny from EST data with widespread gene duplication,” *BMC Evolutionary Biology*, vol. 7, no. Suppl 1, p. S3, 2007.

[256] J. G. Burleigh, M. S. Bansal, O. Eulenstein, S. Hartmann, A. Wehe, and T. J. Vision, “Genome-Scale Phylogenetics: Inferring the Plant Tree of Life from 18,896 Gene Trees,” *Systematic Biology*, vol. 60, no. 2, pp. 117–125, 2010.

[257] R. Chaudhary, B. Boussau, J. G. Burleigh, and D. Fernández-Baca, “Assessing Approaches for Inferring Species Trees from Multi-Copy Genes,” *Systematic Biology*, vol. 64, no. 2, pp. 325–339, 2014.

[258] L. Arvestad, J. Lagergren, and B. Sennblad, “The Gene Evolution Model and Computing Its Associated Probabilities,” *Journal of the ACM*, vol. 56, no. 2, p. 7, 2009.

[259] D. Emms and S. Kelly, “STAG: Species Tree Inference from All Genes,” *bioRxiv*, p. 267914, 2018, available at https://dx.doi.org/10.1101/267914.

[260] G. Ganapathy, B. Goodson, R. Jansen, H.-s. Le, V. Ramachandran, and T. Warnow, "Pattern Identification in Biogeography," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 3, no. 4, pp. 334–346, 2006.

[261] C. Zhang, M. Rabiee, E. Sayyari, and S. Mirarab, "ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees," *BMC Bioinformatics*, vol. 19, no. 6, p. 153, 2018.

[262] A. Markin and O. Eulenstein, "Quartet-Based Inference Methods are Statistically Consistent Under the Unified Duplication-Loss-Coalescence Model," *arXiv*, p. 2004.04299, 2020, available at https://arxiv.org/abs/2004.04299.

[263] R. Dondi, M. Lafond, and C. Scornavacca, "Reconciling multiple genes trees via segmental duplications and losses," *Algorithms for Molecular Biology*, vol. 14, p. 7, 2019.

[264] I. Ullah, P. Parviainen, and J. Lagergren, "Species Tree Inference Using a Mixture Model," *Molecular Biology and Evolution*, vol. 32, no. 9, pp. 2469–2482, 2015.

[265] C. Zhang, C. Scornavacca, E. K. Molloy, and S. Mirarab, "ASTRAL-Pro: quartet-based species tree inference despite paralogy," *bioRxiv*, p. 874727, 2019, available at https://dx.doi.org/10.1101/2019.12.12.874727.

[266] V. P. and W. T., "Enhancing Searches for Optimal Trees Using SIESTA," in *Comparative Genomics. RECOMB-CG 2017. Lecture Notes in Computer Science*, J. Meidanis and L. Nakhleh, Eds. Cham, Switzerland: Springer, 2017, vol. 10562, pp. 232–255.

[267] J. Gatesy, R. W. Meredith, J. E. Janecka, M. P. Simmons, W. J. Murphy, and M. S. Springer, "Resolution of a concatenation/coalescence kerfuffle: partitioned coalescence support and a robust family-level tree for Mammalia," *Cladistics*, vol. 33, no. 3, pp. 295–332, 2017.

[268] J. Gatesy, D. B. Sloan, J. M. Warren, R. H. Baker, M. P. Simmons, and M. S. Springer, "Partitioned coalescence support reveals biases in species-tree methods and detects gene trees that determine phylogenomic conflicts," *Molecular Phylogenetics and Evolution*, vol. 139, p. 106539, 2019.

[269] N. L. Du P., Ogilvie H.A., "Unifying Gene Duplication, Loss, and Coalescence on Phylogenetic Networks," in *Bioinformatics Research and Applications. ISBRA 2019. Lecture Notes in Computer Science*, L. M. Cai Z., Skums P., Ed. Cham, Switzerland: Springer, 2019, vol. 11490, pp. 40–51.

[270] S. Höhna, M. J. Landis, T. A. Heath, B. Boussau, N. Lartillot, B. R. Moore, J. P. Huelsenbeck, and F. Ronquist, "RevBayes: Bayesian Phylogenetic Inference Using Graphical Models and an Interactive Model-Specification Language," *Systematic Biology*, vol. 65, no. 4, pp. 726–736, 2016.

[271] Y. Wang, H. A. Ogilvie, and L. Nakhleh, "Practical Speedup of Bayesian Inference of Species Phylogenies by Restricting the Space of Gene Trees," *Molecular Biology and Evolution*, 2020, msaa045.

APPENDIX A: LIST OF ACRONYMS

AD average distance 15, 33, 64

AGID average gene tree internode distance 27, 63

BME balanced minimum evolution 27

CA-ML concatenation analysis with maximum likelihood 25, 31, 53, 86

DLCoal Duplication, Loss, and Coalescence 5, 16, 114

DP dynamic programming 4, 113, 140

DTM disjoint tree merger 4, 54, 85, 140

EPS effective population size 13, 17, 64, 125

FN false negative 9

FP false positive 9, 32

GDL gene duplication and loss 2, 15, 16, 112, 120, 140

GM General Markov 21, 23

GTEE gene tree estimation error 28, 29, 31, 65, 113, 141

GTP Gene Tree Parsimony 112

GTR Generalized Time Reversible 1, 20, 22, 25, 61, 96

HGT horizontal gene transfer 12, 13, 113, 141

i.i.d. independently and identically distributed 13, 15, 113

ILS incomplete lineage sorting 2, 15, 16, 25, 26, 31, 64, 97, 112

JC Jukes-Cantor 1, 21, 23

MBSS maximum bipartition support supertree 11

MCFD	mutually consistent full differentiation 115
ML	maximum likelihood 1, 22, 24, 25, 34, 62, 105, 126, 141
MLBS	multi-locus bootstrapping 35
MLC	Multi-Locus Coalescent 18
MQSS	maximum quartet support supertree 10, 27, 30, 113
MRCA	most recent common ancestor 7, 16, 121
MSA	multiple sequence alignment 2, 22, 23, 25, 27, 34, 65, 105, 126
MSC	Multi-Species Coalescent 2, 10, 13, 25, 26, 27, 29, 31, 53, 96, 140
NCM	No Common Mechanism 19, 113
NJ	Neighbor Joining 23, 27, 55, 86
QFM	Quartet Fiduccia Mattheyses 10, 30
RF	Robinson-Foulds 8, 15, 18, 34, 59, 94, 114
RFS	Robinson-Foulds supertree 11, 116, 140
SMC	strict majority consensus 41, 51
SRH	stationary, reversible, and homogenous 1, 19, 22, 23, 105
SVD	singular value decomposition 29
UCE	ultraconserved element 12, 51
WF	Wright-Fisher 16

APPENDIX B: LIST OF TERMS