TreeSolve: Rapid Error-Correction of Microbial Gene Trees

Misagh Kordi and Mukul S. Bansal(✉)

Department of Computer Science and Engineering,
University of Connecticut, Storrs, USA
{misagh.kordi,mukul.bansal}@uconn.edu

Abstract. Gene tree reconstruction is an important problem in phylogenetics. However, gene sequences often lack sufficient information to confidently distinguish between competing gene tree topologies. To overcome this limitation, the best gene tree reconstruction methods use a known species tree topology to guide the reconstruction of the gene tree. While such *species-tree-aware* gene tree reconstruction methods have been repeatedly shown to result in vastly more accurate gene trees, the most accurate of these methods often have prohibitively high computational costs.

In this work, we introduce a highly computationally efficient and robust species-tree-aware method, named *TreeSolve*, for microbial gene tree reconstruction. TreeSolve works by collapsing weakly supported edges of the input gene tree, resulting in a non-binary gene tree, and then using new algorithms and techniques to optimally resolve the non-binary gene trees with respect to the given species tree in an appropriately and dynamically constrained search space. Using thousands of real and simulated gene trees, we demonstrate that TreeSolve significantly outperforms the best existing species-tree-aware methods for microbes in terms of accuracy, speed, or both. Crucially, TreeSolve also implicitly keeps track of multiple optimal gene tree reconstructions and can compute either a single best estimate of the gene tree or multiple distinct estimates. As we demonstrate, aggregating over multiple gene tree candidates helps distinguish between correct and incorrect parts of an error-corrected gene tree. Thus, TreeSolve not only enables rapid gene tree error-correction for large gene trees without compromising on accuracy, but also enables accounting of inference uncertainty.

Keywords: Phylogenetics · Microbial evolution · Gene tree reconstruction · Horizontal gene transfer

1 Introduction

One of the most fundamental tasks in studying gene family evolution is the construction of a *gene tree* showing the evolutionary relationships among individual genes from that gene family. However, it is well known that gene trees can be

C. Martín-Vide et al. (Eds.): AlCoB 2020, LNBI 12099, pp. 125–139, 2020.
https://doi.org/10.1007/978-3-030-42266-0_10

very hard to reconstruct accurately and there is often considerable uncertainty in gene tree topologies reconstructed using gene sequences alone [2,8–10]. To address the problem of gene tree error, many *species-tree-aware* methods have been developed for reconstructing or error-correcting gene trees. These methods make use of a known species tree and a phylogenetic reconciliation model that makes it possible to extract topological information from the species tree and use it to guide gene tree inference. In this work, we focus specifically on the reconstruction of microbial gene trees, where the relevant phylogenetic reconciliation model is the Duplication-Transfer-Loss (DTL) reconciliation which models the evolution of gene trees within species trees through speciation, gene duplication, gene loss, and horizontal gene transfer. Given its importance to understanding microbial evolution, the DTL reconciliation problem has been widely studied, e.g., [1,3,4,7,11–16], and all existing species-tree-aware methods for microbial gene trees are based on DTL reconciliation or its variants. Existing species-tree-aware methods for microbial gene trees include AnGST [3], MowgliNNI [9], ALE [15], PrIME-DLTRS [12], TreeFix-DTL [2], TERA [11], and ecceTERA [5]. Amongst all these methods, TreeFix-DTL [2] and ecceTERA [5] have been shown to be among the most accurate. Both TreeFix-DTL and ecceTERA are *gene tree error-correction* methods that take as input a previously reconstructed sequence-only gene tree and error-correct it based on a given species tree. Note that ecceTERA also implements the amalgamation-based algorithm implemented in TERA [11]; however, in this manuscript, ecceTERA refers only to the implementation of the gene tree resolution algorithm from [5].

In this work, we introduce a new species-tree-aware method, *TreeSolve* (portmanteau of *Tree* and *Resolve*), for error-correction of microbial gene trees that significantly outperforms the best existing methods in terms of accuracy, speed, or both. TreeSolve builds upon two key ideas already used for microbial gene tree error-correction and combines and extends them in novel ways. The first of these two keys ideas is to handle gene tree uncertainty by collapsing all weakly supported edges in the input sequence-based gene tree, resulting in a non-binary gene tree, and then optimally resolving this non-binary gene trees by reconciling to the given species tree, e.g., [5,7,17]. The second key idea is the consideration of gene tree bootstraps or other replicates to constrain the search space for the final gene tree to only a biologically meaningful subset of the full search space [3,11,15]. While both of these ideas have been separately used before, TreeSolve combines and extends them to achieve improved speed and accuracy. Specifically, TreeSolve collapses weakly supported edges of the input gene tree, resulting in a non-binary gene tree, and then uses new algorithms and techniques to optimally resolve the non-binary gene trees with respect to the given species tree in a constrained search space defined by a collection of bootstrap/replicate gene tree. An important novel aspect of our algorithm is that it is *self-adaptive* in that it can automatically increase or decrease the search space by considering only those clades that appear in at least a certain fraction of the bootstrap/replicate gene trees (by default, the considered clades should appear in at least one of the bootstrap/replicate gene trees). This self-adaptability is

required because, even with the constraints imposed by the gene tree bootstraps/replicates, the number of optimal resolutions can grow exponentially in the degree and number of non-binary nodes in the given non-binary gene tree. By dynamically increasing or decreasing the minimum support value required for the clades considered, the algorithm is guaranteed to be very efficient even on very large and highly non-binary gene trees while still maintaining its accuracy. Another key strength of TreeSolve is that it implicitly keeps track of multiple, equally optimal, gene tree resolutions; it can either output a single best estimate of the gene tree or it can output multiple distinct gene tree candidates ordered by their average bootstrap/replicate support values.

We compared the accuracy and runtime of TreeSolve against the two most accurate gene tree error-correction methods for microbial gene trees, TreeFix-DTL [2] and ecceTERA [5], using an extensive experimental study with thousands of real and simulated gene trees. TreeFix-DTL has been previously demonstrated to have greater accuracy than AnGST and MowgliNNI [2], and ecceTERA demonstrated to have either greater or comparable accuracy to ALE, TERA, and PrIME-DLTRS [5,11]). Furthermore, ecceTERA is among the fastest species-tree-aware methods currently available for microbial gene trees, and it is also the method conceptually most similar to TreeSolve. Our results demonstrate that (i) TreeSolve is orders of magnitude faster and far more scalable than TreeFix-DTL, while matching or exceeding it in accuracy on larger gene trees, (ii) TreeSolve is significantly more accurate than ecceTERA and has comparable running times, (iii) the self-adaptive algorithm implemented in TreeSolve is highly scalable and efficient and can be easily applied to large genome-scale datasets and gene trees having many hundreds of leaves, and (iv) aggregating over multiple gene tree candidates output by TreeSolve helps distinguish between correct and incorrect branches of an error-corrected gene tree. An implementation of TreeSolve is available from https://compbio.engr.uconn.edu/software/TreeSolve/.

2 Definitions and Preliminaries

We follow basic definitions and notation from [1] and [7]. Given a tree T, we denote its node, edge, and leaf sets by $V(T)$, $E(T)$, and $Le(T)$ respectively.

If T is rooted, the root node of T is denoted by $rt(T)$, the parent of a node $v \in V(T)$ by $pa_T(v)$, its set of children by $Ch_T(v)$, and the (maximal) subtree of T rooted at v by $T(v)$. The set of *internal nodes* of T, denoted $I(T)$, is defined to be $V(T) \setminus Le(T)$. For a rooted tree T, we define $\leq_T$ to be the partial order on $V(T)$ where $x \leq_T y$ if y is a node on the path between $rt(T)$ and x. The partial order $\geq_T$ is defined analogously, i.e., $x \geq_T y$ if x is a node on the path between $rt(T)$ and y. We say that y is an *ancestor* of x, or that x is a *descendant* of y, if $x \leq_T y$ (note that, under this definition, every node is a descendant as well as ancestor of itself). We say that x and y are *incomparable* if neither $x \leq_T y$ nor $y \leq_T x$. Given a non-empty subset $L \subseteq Le(T)$, we denote by $lca_T(L)$ the last common ancestor (LCA) of all the leaves in L in tree T.

A rooted tree is *binary* if all of its internal nodes have exactly two children, and *non-binary* otherwise. An *internal edge* is an edge whose end points are both internal nodes in the tree. An internal edge $(x, pa_T(x))$ in tree T can be *contracted* by removing $(x, pa_T(x))$ and creating new edges joining $pa_T(x)$ with $Ch_T(x)$, thereby yielding a new tree distinct from T. We say that a tree T' is a *binary resolution* of T if T' is binary and T can be obtained from T' by contracting some (zero or more) internal edges. We denote by $\mathcal{BR}(T)$ the set of all binary resolutions of a rooted non-binary tree T. Given any node x from T, we define the *out-degree* of x to be the number of children of x.

For a rooted tree T each node $v \in V(T)$, the *clade* $C_T(v)$ is defined to be the set of all leaf nodes in $T(v)$; i.e. $C_T(v) = Le(T(v))$. We denote the set of all clades of a rooted tree T by $Clade(T)$. This concept can be extended to unrooted trees as follows. Suppose T is an unrooted tree. Each edge $(u, v) \in E(T)$ defines a partition of the leaf set of T into two disjoint subsets $Le(T_u)$ and $Le(T_v)$, where T_u is the subtree containing node u and T_v is the subtree containing node v, obtained when edge (u, v) is removed from T. We call $Le(T_u)$ and $Le(T_v)$ the *clusters* of T induced by edge (u, v), and denote the set of all clusters in an unrooted tree T by $Cluster(T)$.

In this work, we will consider both rooted and unrooted trees. However, unless otherwise specified, the term *tree* refers to a rooted tree.

A *species tree* is a tree that depicts the evolutionary relationships of a set of species. Given a gene family from a set of species, a *gene tree* is a tree that depicts the evolutionary relationships among the sequences encoding only that gene family in the given set of species. Gene trees may be either binary or non-binary while the species tree is always assumed to be binary. Throughout this work, we denote the gene tree and species tree under consideration by G and S, respectively. If G is restricted to be binary we refer to it as G^B and as G^N if it is restricted to be non-binary. We assume that each leaf of the gene tree is labeled with the species from which that gene was sampled. This labeling defines a *leaf-mapping* $\mathcal{L}_{G,S}\colon Le(G) \rightarrow Le(S)$ that maps a leaf node $g \in Le(G)$ to that unique leaf node $s \in Le(S)$ that has the same label as g. Note that gene trees may have more than one gene sampled from the same species.

Reconciliation and DTL-scenarios. A binary gene tree can be reconciled with a species tree by mapping the gene tree into the species tree. A Duplication-Transfer-Loss scenario (DTL-scenario) [1,16] for G^B and S characterizes the mappings of G^B into S that constitute a biologically valid reconciliation. Essentially, DTL-scenarios map each gene tree node to a unique species tree node and designate each gene tree node as representing either a speciation, duplication, or transfer event. A formal definition of DTL-scenario appears in [1]. DTL-scenarios correspond naturally to reconciliations and it is straightforward to infer the reconciliation of G^B and S implied by any DTL-scenario. Given a DTL-scenario, one can directly count the number of duplications, transfers, and losses invoked by the corresponding reconciliation [1].

Let P_Δ, P_Θ, and P_{loss} denote the non-negative costs associated with duplication, transfer, and loss events, respectively. The *reconciliation cost* of a DTL-

scenario is defined to be the total cost of all duplication, transfer, and loss events invoked by that DTL-scenario. A most parsimonious reconciliation is one that has minimum reconciliation cost.

Definition 1 (MPR). *Given G^B and S, along with P_Δ, P_Θ, and P_{loss}, a* most parsimonious reconciliation (MPR) *for G^B and S is a DTL-scenario with minimum reconciliation cost.*

Optimal Gene Tree Resolution. TreeSolve works by first converting the given binary gene tree into a non-binary gene tree by collapsing weakly supported edges (based on a user-provided threshold), and then optimally resolving this non-binary gene tree based on the species tree under appropriate topological constraints. A closely related problem formulation that has been previously studied is that of optimal gene tree resolution (OGTR) under DTL reconciliation [6,7]. In the OGTR problem, given non-binary gene tree G^N and a species tree, one must find a binary resolution G^B of G^N such that an MPR of G^B with S has smallest reconciliation cost. Moreover, since there may be more than one optimal binary resolution of G^N, the desired formulation of the problem is to find *all* optimal resolutions of G^N. This leads to the following computational problem [7].

Problem 1 (OGTR-All). *Given G^N and S, along with P_Δ, P_Θ, and P_{loss}, the* All Optimal Gene Tree Resolutions (OGTR-All) *problem is to compute the set $\mathcal{OR}(G^N)$ of all optimal binary resolutions of G^N such that, for any $G^B \in \mathcal{OR}(G^N)$, an MPR of G^B and S has the smallest reconciliation cost among all gene trees in $\mathcal{BR}(G^N)$.*

The OGTR-All problem is known to be NP-hard [6] (even for computing a single optimal resolution), and existing algorithms are limited to solving instances in which the maximum out-degree in G^N is small [7].

Constrained Optimal Gene Tree Resolution. In addition to its very high computational time complexity, which greatly limits its applicability, the OGTR-All problem ignores sequence information and is therefore prone to over-fitting the gene tree to the species tree. TreeSolve addresses both these limitations by constraining the set of binary resolutions of G^N that can be considered. Specifically, TreeSolve allows all binary resolutions that are supported by the sequence data and disallows those that are unsupported. To achieve this goal TreeSolve solves a constrained version of the OGTR-All problem in which, in addition to G^N and S, we take as input a set of unrooted gene trees that define constraints on the set of binary resolutions of G^N. The set of unrooted gene trees used should represent a sample of gene tree topologies supported by the sequence data and can be easily obtained by either computing bootstrap replicates or sampling from the posterior distribution in a Bayesian analysis.

More formally, let $\mathbb{B} = \{B_1, B_2, \ldots, B_b\}$ denote a sample of b unrooted gene trees. Then, we define the cluster set of $\mathbb{B}$ to be: $\mathit{Cluster}(\mathbb{B}) = \bigcup_{i=1}^{b} \mathit{Cluster}(B_i)$. This set of clusters is used to define the constrained set of binary resolutions as follows.

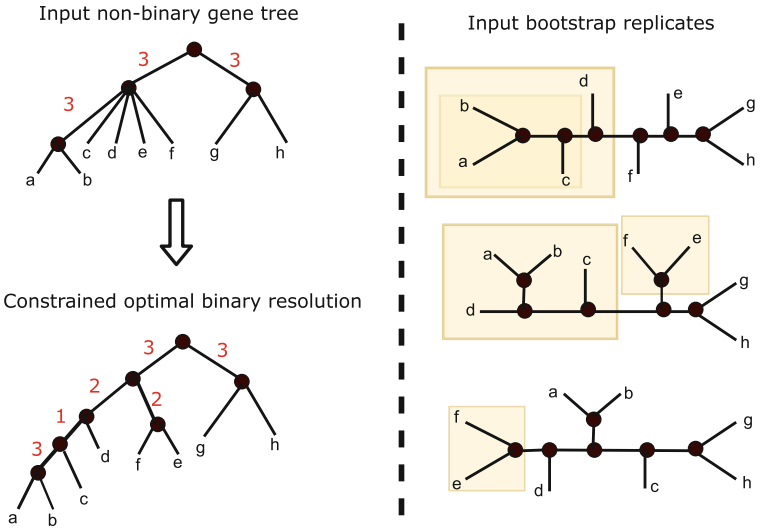

Fig. 1. Constrained binary resolutions. Given the (rooted) non-binary gene tree on the top left and the three (unrooted) bootstrap replicate gene trees on the right, the figure shows a possible constrained binary resolution of the non-binary gene tree. Note that each new clade in the binary resolution appears as a cluster in at least one of the three bootstrap replicate trees. These clusters are highlighted using the yellow boxes. Each internal edge in the gene trees is labeled by its branch support (number in red), i.e., the number of bootstrap replicates that support that branch. In this example, the constrained binary resolution happens to be a constrained *optimal* binary resolution since no other constrained binary resolution has higher average branch support. (Color figure online)

Definition 2 (Constrained binary resolution). *Given* $\mathbb{B}$ *and a non-binary tree* T*, we say that* T' *is a* constrained binary resolution *of* T *(with respect to* $\mathbb{B}$*), if* $T' \in \mathcal{BR}(T)$ *and* $Clade(T') \subseteq Cluster(\mathbb{B})$*. We denote by* $\mathcal{CBR}(T)$ *the set of all constrained binary resolutions of a rooted non-binary tree* T*.*

The idea of a constrained binary resolution is illustrated in Fig. 1. We can now state the constrained optimal gene tree resolution problem.

Problem 2 (C-OGTR). *Given* G^N*,* S*, and* $\mathbb{B}$*, along with* P_Δ*,* P_Θ*, and* P_{loss}*, the* All Constrained Optimal Gene Tree Resolutions (C-OGTR) *problem is to compute the set* $\mathcal{COR}(G^N)$ *of all optimal constrained binary resolutions of* G^N *such that, for any* $G^B \in \mathcal{COR}(G^N)$*, an MPR of* G^B *and* S *has the smallest reconciliation cost among all gene trees in* $\mathcal{CBR}(G^N)$*.*

Note: To ensure that a solution always exists to the C-OGTR problem, Tree-Solve includes the original binary gene tree from which G^N is obtained in the set $\mathbb{B}$. This ensures, that a constrained binary resolution of G^N always exists.

We also define a variant of the problem above that only seeks to find a single optimal reconciliation with highest average clade support.

Problem 3 (C-OGTR-Best). *Given G^N, S, and $\mathbb{B}$, along with P_Δ, P_Θ, and P_{loss}, the* Best Constrained Optimal Gene Tree Resolutions (C-OGTR-Best) *problem is to compute a tree $G^B \in \mathcal{CBR}(G^N)$ such that the total number of occurrences in $\mathbb{B}$ of all clades in G^B is the largest among all trees in $\mathcal{CBR}(G^N)$.*

Note that TreeSolve does not directly solve the *C-OGTR* and *C-OGTR-Best* problems. Rather, for improved efficiency and accuracy, TreeSolve solves variants of these problems where *Cluster*($\mathbb{B}$) is further restricted to only contain those clusters that are present in at least a certain number, *minSup*, of the samples in $\mathbb{B}$, where *minSup* is updated dynamically during the search. Furthermore, TreeSolve maintains ordered lists of binary resolutions at each step sorted by average support value (i.e., by the total number of occurrences in $\mathbb{B}$ of all clades in that binary resolution).

3 Algorithmic Overview

Our algorithms for the *C-OGTR* and *C-OGTR-Best* problems leverage the dynamic programming algorithm for the *OGTR-All* problem described in [7]. The primary difference is that the new algorithms limit the possible binary resolutions considered at each non-binary node to those that can be constructed from the clusters available in *Cluster*($\mathbb{B}$). Further technical details are omitted for brevity. Here, we describe how solutions for *C-OGTR*/*C-OGTR-Best* are used within TreeSolve as part of the larger self-adaptive approach and optimal resolution ordering.

TreeSolve's Self-adaptive Algorithm. Note that, despite the restriction on permitted resolutions imposed by $\mathbb{B}$, the total number of constrained optimal binary resolutions can be exponential in the number of non-binary nodes of G^N as well as its maximum out-degree. To address this limitation, TreeSolve employs a novel self-adaptive approach to limit the number of binary resolutions considered at each non-binary node. To describe the self-adaptive approach, we need some additional definitions and notation. We first define an upper bound, denoted U, on the total number of binary resolutions considered by TreeSolve during any step in its execution. For example, for all the experimental results presented in the next section, we assigned $U = 25000$. We also define the following:

$$\mathit{Cluster}(\mathbb{B}, \mathit{minSup}) = \{x \in \mathit{Cluster}(\mathbb{B}) \mid x \text{ appears in at least } \mathit{minSup} \text{ trees from } \mathbb{B}\}.$$

Finally, define $N(g, \mathit{minSup})$ to be the number of distinct binary resolutions of the non-binary node $g \in G^N$ permitted by the cluster set *Cluster*($\mathbb{B}$, *minSup*). For each non-binary node $g \in G^N$ independently, TreeSolve computes a value for *minSup* for which $N(g, \mathit{minSup}) \leq U$ but $N(g, \mathit{minSup}-1) > U$. This can be accomplished efficiently through a binary-search in the range $[1, |\mathbb{B}|]$. Thus, at each non-binary node of the gene tree, we limit the total number of resolutions considered to at most U of the most highly supported ones.

Ordering of Binary Resolutions by Average Clade Support. In addition to its use for limiting the number of possible resolutions at each non-binary node, TreeSolve also uses the upper bound U to bound the total number of resolutions considered at the subtree rooted at each node of the gene tree. In other words, TreeSolve executes a variant of the algorithm for *C-OGTR* that always limits the total number of resolutions of the subtree $G^N(g)$ stored at any node g of the gene tree to U. In particular, at each node $g \in G^N$ the algorithm only stores up to the U best (in terms of average clade support) resolutions for the subtree $G^N(g)$ encountered during the search, ordered by their average clade support. We denote this ordered list of the U best resolutions for the subtree $G^N(g)$ by $\mathcal{ORV}(g)$ (for optimal resolution vector). Note that each resolution stored in $\mathcal{ORV}(g)$ also has an associated average clade support value stored along with it. Next, we describe how each $\mathcal{ORV}(\cdot)$ is computed as part of the bottom-up dynamic programming traversal of G^N. We first need some additional notation. Given any binary or non-binary node $g \in G^N$, define the set of nearest non-binary descendants of g, denoted $\mathcal{N}(g)$, to be $\{h \in V(G^N(g)) \setminus \{g\} \mid h$ is non-binary and no other non-binary nodes exist on the path from g to $h\}$. Note that $\mathcal{N}(g)$ may be empty.

Consider any binary or non-binary node $g \in G^N$. If all nodes in the subtree $G^N(g)$ are binary then there is only one possible resolution (i.e., the current resolution). If $\mathcal{N}(g) = \emptyset$ but g itself is non-binary then we apply the self-adaptive approach described above and compute up to U binary resolutions of $G^N(g)$. These resolutions are then sorted according to decreasing average clade support (based on the trees in $\mathbb{B}$) and stored as $\mathcal{ORV}(g)$. If $\mathcal{N}(g) \neq \emptyset$ and g is binary, then $\mathcal{ORV}(g)$ can be computed by suitably combining the vectors $\mathcal{ORV}(h)$, for each $h \in \mathcal{N}(g)$, already computed in previous steps of the algorithm. Observe that each combination of resolutions from the $\mathcal{ORV}(h)$'s, across all $h \in \mathcal{N}(g)$, yields a permitted resolution for the subtree $G^N(g)$. Since each $\mathcal{ORV}(h)$ is in sorted order and each resolution is associated with its average clade support value, computing the U best resolutions for $G^N(g)$, i.e., computing $\mathcal{ORV}(g)$, can be accomplished by performing a merge-like procedure (from merge sort) on the $\mathcal{ORV}(h)$'s to identify just the U best resolutions for $G^N(g)$. The remaining case, where $\mathcal{N}(g) \neq \emptyset$ and g is non-binary can be handled similarly by considering the sorted list of permitted resolutions for node g together with the $\mathcal{ORV}(h)$'s.

Computing Only a Single Best Resolution. By default, TreeSolve computes a sorted list of up to U (where $U = 25000$ in all our experiments) distinct best resolutions of the initial non-binary gene tree. However, in many applications, only a single best estimate of the error-corrected gene tree may be required. Indeed, most existing species-tree-aware methods for microbial gene tree error-correction, including TreeFix-DTL and ecceTERA, only compute a single best gene tree. It is easy to see that the first tree output by TreeSolve corresponds to this best tree, i.e., with highest average clade support. However, if only the best solution was required, TreeSolve could make use of the simpler *C-OGTR-Best* problem formulation, instead of the *C-OGTR* problem as described above. Solving the *C-OGTR-Best* problem is simpler and more efficient than the *C-OGTR*

problem (though still potentially exponential). Specifically, to only compute the resolution with highest average clade support, we need not maintain $\mathcal{ORV}(\cdot)$ vectors and only need to save the best resolution corresponding to each subproblem $c(g, s)$.

4 Experimental Evaluation

Simulated and Real Datasets Used in the Analysis. To evaluate the performance our new approach, we used the large simulated dataset of 2400 gene tree/species tree pairs on 50 taxa used in [2] to evaluate the accuracy of TreeFix-DTL. These 2400 gene trees represent 24 categories (each with 100 gene trees) that capture a wide range of evolutionary scenarios. Specifically, the datasets represent all combinations of (i) low, medium, and high rates of duplication, transfer, and loss events, (ii) four different sequence mutation rates (rates 1, 3, 5 and 10), and (iii) normal (333 amino acids) and short (173 amino acids) sequence lengths; further details on the construction of datasets are available in [2]. For each of the 2400 gene tree/species tree pairs in this dataset, we have available the true (simulated) gene tree and species tree, the reconstructed maximum likelihood gene tree (constructed using RAxML on sequence data simulated down the true gene tree), and 100 bootstrap replicates computed during the execution of RAxML. The 24 categories in this dataset span a wide range of gene tree sizes, event rates, and error rates: specifically, the average leaf set size of the low, medium, and high DTL gene trees are 52.3, 70.4, and 91.3, respectively; the average count of evolutionary events (duplications, transfers, and losses) for the low, medium, and high DTL gene trees are 5.5, 10.6, and, 18.8, respectively, with transfers constituting roughly half of each count; and baseline RAxML error rates (in terms of NRFD, as defined below) ranging from a low of less than 6% to a high of almost 18%.

To further test the scalability and accuracy of TreeSolve on large datasets, we used a dataset of 200 gene tree/species tree pairs on 200 taxa also used in [2]. These 200 gene trees represent 2 distinct categories corresponding to normal sequence length (333 amino acids), a medium rate of DTL, and mutation rates 1 and 5.

In addition to the simulated dataset above, we also used a real biological dataset of over 4700 gene trees from 100 predominantly prokaryotic species [3]. We use this dataset to demonstrate scalability and the impact of applying TreeSolve in practice.

Experimental Setup. We evaluated the accuracy and runtime of TreeSolve, TreeFix-DTL, and ecceTERA on each dataset described above. TreeSolve and ecceTERA both take as input a gene tree with support values on its edges, a support cutoff threshold to collapse edges with weak support (thereby producing a non-binary gene tree), and a rooted species tree. In addition, TreeSolve also takes as input the set $\mathbb{B}$ of bootstrap or other samples based on which the gene tree edge support values were computed. TreeFix-DTL takes as input a

maximum-likelihood (e.g., RAxML) gene tree, the corresponding sequence alignment, and a rooted species tree. We used default event cost values of 1, 2, and 3, for losses, duplications, and transfers, respectively, for TreeSolve, ecceTERA, and TreeFix-DTL (all three programs use these same event costs by default). The rooted gene trees given as input to TreeSolve and ecceTERA were obtained by rooting each reconstructed RAxML gene tree at an edge that minimized its DTL reconciliation cost. To create the non-binary gene trees for TreeSolve and ecceTERA, we tried two different support cutoff thresholds: 50% and 90%. Note that using higher bootstrap cutoff values results in more non-binary (i.e., more unresolved) gene trees as more edges are collapsed. We observed that both ecceTERA and TreeSolve performed significantly better when the higher cutoff threshold of 90% was used; specifically, the average error-rate in terms of NRFD (defined below) across the 24 baseline simulated datasets decreased from 7.1% to 5.7% for ecceTERA and from 7.7% to 4.85% for TreeSolve. Thus, we fixed the cutoff threshold to 90% for all our experiments with ecceTERA and TreeSolve, including those with real biological data. For each simulated and real gene tree, 100 bootstrap replicates obtained through RAxML were used to define the corresponding set $\mathbb{B}$ for TreeSolve.

To measure gene tree accuracy, we used the (unrooted) normalized Robinson-Foulds distance (NRFD) against the true gene tree; for any reconstructed gene tree, the NRFD shows the percentage of splits in that gene tree that do not appear in the corresponding true gene tree. Finally, when evaluating the accuracy of TreeSolve, unless otherwise stated, we use only the best (ie., first) resolution computed.

Basic Statistics on Datasets. For the 24 baseline simulated datasets, the average leaf set size of the low, medium, and high DTL gene trees was 52.33, 70.37, and 91.26, respectively. Upon collapsing weakly supported edges with the 90% cutoff threshold, we found that the average number of non-binary nodes and average of maximum out-degrees across all 24 baseline simulated datasets were 10.9 and 8.2, respectively, with the highest averages observed to be 15.9 (for sequence length 173, rate-10, high DTL) and 20.58 (for sequence length 173, rate-1, high DTL), respectively.

For the larger 200-taxon simulated datasets, the average number of non-binary nodes was 39.5 and the average of maximum out-degrees was 6.7.

For the real dataset of 4736 gene trees, we found that 4419 became non-binary at a 90% bootstrap cutoff threshold. For these 4419 non-binary gene trees, the average leaf set size was 36.1, the largest leaf set size was 600, and the average number of non-binary nodes and average of maximum out-degrees were 3.35 and 21.14, respectively.

4.1 Results

Simulated Datasets Results. We first compared the accuracies of TreeSolve, ecceTERA, and TreeFix-DTL on the 24 baseline simulated datasets. These results are shown in Fig. 2. As the figure shows, *TreeSolve* results in significantly more accurate gene tree resolutions than *ecceTERA* in 19 out of the 24

datasets (and across all high DTL datasets), while TreeFix-DTL outperforms both ecceTERA and TreeSolve on all 24 datasets. As we discuss later, this improved accuracy of TreeFix-DTL comes at the expense of orders of magnitude greater running time. The average normalized Robinson-Foulds distances (NRFD) for RAxML, ecceTERA, TreeSolve, and TreeFix-DTL are 7.4%, 3.9%, 3.1%, and 1.86%, respectively, across the 12 normal sequence-length datasets, and 12.5%, 7.6%, 6.6%, and 3.8%, respectively, across the 12 short sequence datasets. As expected, all three species-tree-aware methods were significantly more accurate than the sequence-only method RAxML, and absolute error rates for all four methods were higher for the short (173) sequence length datasets than for the normal (333) length datasets. Interestingly, we observed that while the accuracies of ecceTERA and TreeFix-DTL consistently worsen with increasing DTL rates, the accuracy of TreeSolve is only slightly affected by DTL rates (Fig. 2). As a result, the accuracy of TreeSolve starts to approach that of TreeFix-DTL on the high DTL datasets. Specifically, the average NRFDs across the high-DTL datasets for TreeFix are 2.7% and 5.1% for the normal and short sequence-length datasets, respectively, while for TreeSolve these numbers are only slightly larger at 3.5% and 6.9%, respectively.

Impact of Gene Tree Size. To study the impact of tree size on the relative accuracies of the three methods, we used the two simulated datasets of 100 gene tree/species tree pairs each on 200 taxa. As expected, TreeSolve continues to significantly outperform ecceTERA on these larger datasets, with an average NRFD of 3.0% for ecceTERA and only 1.8% for TreeSolve. More significantly, we find that TreeSolve slightly outperforms TreeFix-DTL on these larger trees, with average NRFD of 1.8% to TreeFix-DTL's 1.85%. This is not entirely surprising, since TreeFix-DTL relies on an iterative local search approach that can become less effective as tree size increases. Thus, TreeSolve can be expected to outperform all other methods for larger gene trees.

Impact of Enumerating Multiple Optimal Resolutions. Recall that a key feature of TreeSolve is that it can compute and output multiple optimal resolutions, ordered by their average support values. To explore the impact of considering multiple optimal resolutions instead of only using the "best" resolution computed through TreeSolve, we computed the false positive and false negative branch rates for the strict consensus of all multiple optimal resolutions computed by TreeSolve. We found that the strict consensus of all optimal resolutions computed by TreeSolve results in a significantly lower false positive rate compared to just using the "best" TreeSolve gene trees across each of the 24 datasets, with an overall average of 3.45% versus 5.85%, respectively. This suggests that the optimal resolutions computed by TreeSolve can be used to distinguish between correct and incorrect gene tree edges. Unsurprisingly, this improvement in the false positive rate comes at the expense of an increased false negative rate, with the average normalized false negative rate over all 24 datasets being 8.5 for the strict consensus of all multiple optimal resolutions computed by TreeSolve. For brevity, detailed results on individual datasets are omitted from this manuscript.

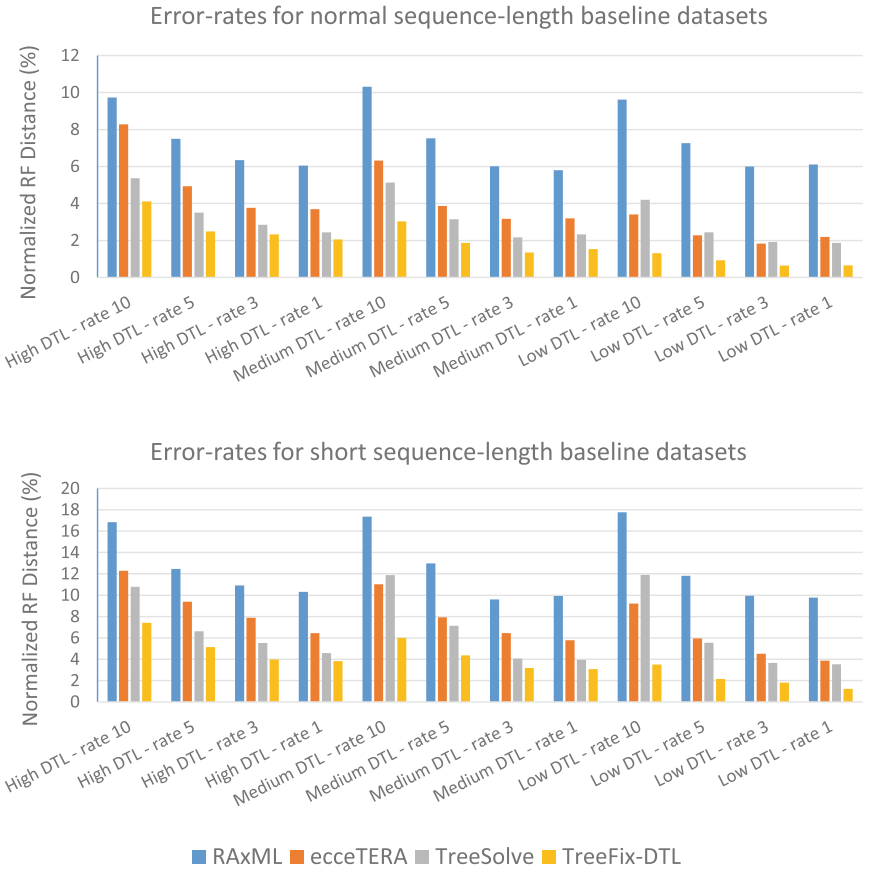

Fig. 2. Accuracy on baseline datasets. Error rates are shown for gene trees inferred using RAxML, ecceTERA, TreeSolve, and TreeFix-DTL on the 12 normal sequence-length (top) and 12 short sequence-length (bottom) simulated datasets.

Running Time and Scalability. Both ecceTERA and TreeSolve required only a few seconds per simulated gene tree. Specifically, the average running time of ecceTERA was 2.9 s per tree across the 24 baseline simulated datasets, and for TreeSolve the corresponding average running time was 10.2 s. TreeFix-DTL was far slower than ecceTERA and TreeSolve, requiring an average of over an hour for each of the trees in these 24 baseline simulated datasets. On the larger 200-taxon simulated datasets, ecceTERA and TreeSolve averaged 2.5 s and 82 s per gene tree, respectively. In contrast, TreeFix-DTL required an average of over 10 h per gene tree. Thus, TreeSolve is almost three orders of magnitude faster than TreeFix-DTL on these larger gene trees while also showing better accuracy. All timed runs were executed using a single core on a commodity Macbook Pro laptop with 16 GB of RAM and a 2.3 GHz Intel i9 CPU.

Results on Real Dataset. We studied the impact of applying ecceTERA, TreeSolve and TreeFix-DTL on a real biological dataset of over 4736 gene trees from 100 predominantly prokaryotic species [3]. We found that 4419 out of the 4736 gene trees became non-binary at the 90% bootstrap cutoff threshold, and ecceTERA and TreeSolve were thus able to error-correct these 4419 trees. Over all gene trees, ecceTERA had an average running time of 15.1 s and a maximum running time of 317 s. TreeSolve had a slightly larger average running time of 71.6 s and a maximum of 1736 s. Note that the largest gene tree in this dataset has 600 leaves. This demonstrates how TreeSolve can be applied to very large gene trees within minutes.

For the 4419 non-binary gene trees, we found that ecceTERA resulted in an average decrease of 26.4% in the reconciliation cost of the error-corrected gene trees. For TreeSolve, this decrease was a much larger 38.5%. The magnitude of decrease in reconciliation cost is a highly imperfect proxy for gene tree accuracy; still, these numbers suggest that TreeSolve is more effective at error-correcting these real gene trees.

In contrast to ecceTERA and TreeSolve, which executed within minutes on even the largest gene trees, TreeFix-DTL required more than a week of running time on each of the larger gene trees in this dataset.

5 Conclusion

In this work, we introduced a new species-tree-aware method, TreeSolve, for error-correcting microbial gene trees. TreeSolve combines new and existing techniques and uses novel algorithms to strike a balance between speed and accuracy. As our extensive experimental analysis demonstrates, TreeSolve significantly outperforms the best existing species-tree-aware methods for microbes in terms of accuracy, speed, or both. TreeSolve is especially effective for error-correction of large gene trees, where it makes it possible to perform speedy error-correction without any compromise on reconstruction accuracy. Furthermore, TreeSolve has the extremely useful ability to compute not just a single best estimate of the error-corrected gene tree but a ranked list of multiple distinct "roughly equally good" candidates. As we show in our experimental study, the resulting ability to aggregate over multiple gene tree candidates helps distinguish between correct and incorrect relationships in an error-corrected gene tree. Overall, TreeSolve has the potential to transform the reconstruction of large microbial gene trees and to increase the robustness of downstream evolutionary inferences by enabling the accounting of gene tree reconstruction uncertainty.

Funding. This work was supported in part by NSF awards MCB 1616514 and IES 1615573 to MSB.

References

1. Bansal, M.S., Alm, E.J., Kellis, M.: Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics **28**(12), 283–291 (2012)
2. Bansal, M.S., Wu, Y.C., Alm, E.J., Kellis, M.: Improved gene tree error correction in the presence of horizontal gene transfer. Bioinformatics **31**(8), 1211–1218 (2015). https://doi.org/10.1093/bioinformatics/btu806
3. David, L.A., Alm, E.J.: Rapid evolutionary innovation during an archaean genetic expansion. Nature **469**, 93–96 (2011)
4. Doyon, J.-P., Scornavacca, C., Gorbunov, K.Y., Szöllősi, G.J., Ranwez, V., Berry, V.: An efficient algorithm for gene/species trees parsimonious reconciliation with losses, duplications and transfers. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 93–108. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16181-0_9
5. Jacox, E., Weller, M., Tannier, E., Scornavacca, C.: Resolution and reconciliation of non-binary gene trees with transfers, duplications and losses. Bioinformatics **33**(7), 980 (2017). https://doi.org/10.1093/bioinformatics/btw778. http://dx.doi.org/10.1093/bioinformatics/btw778
6. Kordi, M., Bansal, M.S.: On the complexity of duplication-transfer-loss reconciliation with non-binary gene trees. IEEE/ACM Trans. Comput. Biology Bioinform. **14**(3), 587–599 (2017)
7. Kordi, M., Bansal, M.S.: Exact algorithms for duplication-transfer-loss reconciliation with non-binary gene trees. IEEE/ACM Trans. Comput. Biology Bioinform. **16**(4), 1077–1019 (2019)
8. Li, H., et al.: Treefam: a curated database of phylogenetic trees of animal gene families. Nucleic Acids Res. **34**(Suppl. 1), D572–D580 (2006)
9. Nguyen, T.H., Doyon, J.-P., Pointet, S., Chifolleau, A.-M.A., Ranwez, V., Berry, V.: Accounting for gene tree uncertainties improves gene trees and reconciliation inference. In: Raphael, B., Tang, J. (eds.) WABI 2012. LNCS, vol. 7534, pp. 123–134. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33122-0_10
10. Rasmussen, M.D., Kellis, M.: A bayesian approach for fast and accurate gene tree reconstruction. Mol. Biol. Evol. **28**(1), 273–290 (2011). https://doi.org/10.1093/molbev/msq189
11. Scornavacca, C., Jacox, E., Szollosi, G.J.: Joint amalgamation of most parsimonious reconciled gene trees. Bioinformatics **31**(6), 841 (2015). https://doi.org/10.1093/bioinformatics/btu728
12. Sjostrand, J., Tofigh, A., Daubin, V., Arvestad, L., Sennblad, B., Lagergren, J.: A bayesian method for analyzing lateral gene transfer. Syst. Biol. **63**(3), 409–420 (2014). https://doi.org/10.1093/sysbio/syu007
13. Stolzer, M., Lai, H., Xu, M., Sathaye, D., Vernot, B., Durand, D.: Inferring duplications, losses, transfers and incomplete lineage sorting with nonbinary species trees. Bioinformatics **28**(18), 409–415 (2012)
14. Szollosi, G.J., Boussau, B., Abby, S.S., Tannier, E., Daubin, V.: Phylogenetic modeling of lateral gene transfer reconstructs the pattern and relative timing of speciations. Proc. Natl. Acad. Sci. USA **109**(43), 17513–17518 (2012). https://doi.org/10.1073/pnas.1202997109
15. Szollosi, G.J., Tannier, E., Lartillot, N., Daubin, V.: Lateral gene transfer from the dead. Syst. Biol. **62**(3), 386 (2013). https://doi.org/10.1093/sysbio/syt003. http://dx.doi.org/10.1093/sysbio/syt003

16. Tofigh, A., Hallett, M.T., Lagergren, J.: Simultaneous identification of duplications and lateral gene transfers. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**(2), 517–535 (2011)
17. Zheng, Y., Zhang, L.: Reconciliation with non-binary gene trees revisited. In: Sharan, R. (ed.) RECOMB 2014. LNCS, vol. 8394, pp. 418–432. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05269-4_33