Bioinformatics, 2022, 1–10
https://doi.org/10.1093/bioinformatics/btab875
Advance Access Publication Date: 3 January 2022
Original Paper

Phylogenetics

Completing gene trees without species trees in sub-quadratic time

Uyen Mai[1] **and Siavash Mirarab**[2,*]

[1]Department of Computer Science and Engineering, University of California San Diego, San Diego, CA 92093, USA and [2]Department of Electrical and Computer Engineering, University of California San Diego, San Diego, CA 92093, USA

*To whom correspondence should be addressed.

Associate Editor: Russell Schwartz

Received on June 23, 2021; revised on November 27, 2021; editorial decision on December 28, 2021; accepted on December 30, 2021

Abstract

Motivation: As genome-wide reconstruction of phylogenetic trees becomes more widespread, limitations of available data are being appreciated more than ever before. One issue is that phylogenomic datasets are riddled with missing data, and gene trees, in particular, almost always lack representatives from some species otherwise available in the dataset. Since many downstream applications of gene trees require or can benefit from access to complete gene trees, it will be beneficial to algorithmically complete gene trees. Also, gene trees are often unrooted, and rooting them is useful for downstream applications. While completing and rooting a gene tree with respect to a given species tree has been studied, those problems are not studied in depth when we lack such a reference species tree.

Results: We study completion of gene trees without a need for a reference species tree. We formulate an optimization problem to complete the gene trees while minimizing their quartet distance to the given set of gene trees. We extend a seminal algorithm by Brodal *et al.* to solve this problem in quasi-linear time. In simulated studies and on a large empirical data, we show that completion of gene trees using other gene trees is relatively accurate and, unlike the case where a species tree is available, is unbiased.

Availability and implementation: Our method, tripVote, is available at https://github.com/uym2/tripVote.

Contact: smirarab@ucsd.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Phylogenetic analyses of genome-wide data (i.e. phylogenomics) typically infer a set of gene trees, each from a different region of the genome (not necessarily a gene), and a species tree, which may be obtained from combining the gene trees. These analyses, in principle, benefit from the size of available data and can have high accuracy. However, phylogenomic datasets are also known to suffer from both partial incompleteness and undiscovered errors (Hosner *et al.*, 2016; Laurin-Lemay *et al.*, 2012; Philippe *et al.*, 2017; Springer and Gatesy, 2018). The preparation of the data for phylogenomic analyses involves many steps, much of them error prone, and these steps can easily miss parts of the sequences. The issue of undetected errors is being increasingly addressed using new methods (e.g. Mai and Mirarab, 2018; Zhang *et al.*, 2020) and simple filtering strategies (Doyle *et al.*, 2015; Hosner *et al.*, 2016; Sayyari *et al.*, 2017). However, by further removing data, many of these methods exacerbate the issue and sometimes have negative effects on tree inference (Mclean *et al.*, 2019; Tan *et al.*, 2015).

There is growing evidence that species tree inference methods are robust to presence of some missing data (Hovmöller *et al.*, 2013; Molloy and Warnow, 2018; Nute *et al.*, 2018; Xi *et al.*, 2016). The incompleteness of gene trees, however, is not just a concern for species tree inference. Gene trees are used for many other analyses, including gene family evolution, functional analyses of proteins, reconstructing ancestral gene content, and dating gene birth. Moreover, many species tree inference methods internally rely on completing gene trees, even if just approximately. For example, ASTRAL completes input gene trees with respect to each other to define a bipartitions set as its search space (Mirarab and Warnow, 2015). Thus, researchers have studied the problem of completing incomplete gene trees using the rest of the data.

The existing gene tree completion methods mostly are based on the same philosophy: that once a species tree is inferred, a gene tree can be completed with respect to that species tree to minimize their distance. What differentiates the methods is what measure of distance they use to achieve that goal. For example, Bayzid and Warnow (2012) use a parsimony framework to minimize deep coalescence. More recently, Christensen *et al.* (2018); Bansal (2018) and later Aiemvaravutigul and Wongwattanakij (2019) show how Robinson and Foulds (1981) (RF) distance can be minimized efficiently.

While these methods are all valuable, they do not directly provide a way to complete gene trees without a species tree. Such a completion may be desired for two reasons. First, completing gene trees with respect to the species tree may artificially reduce the amount of observed discordance. For example, if we use the species tree from the plant dataset of OneKP Initiative (2019), to complete gene trees by minimizing the RF distances, the mean normalized RF distance of the gene trees to the species tree drops by 8%, meaning that the observed discordance paradoxically reduces as a result of gene tree completion. This level of discordance leads to an increase in estimated coalescent unit branch lengths of 0.26 on average. Thus, the added taxa are artificially less discordant with the species tree than the backbone species. Second, species trees are not always available where gene trees are. For example, part of the ASTRAL algorithm completes gene trees before the species tree is inferred.

We formulate gene tree completion without species trees as follows: Given a set of gene trees, complete each gene tree with respect to the other gene trees such that its overall distance to the other trees is minimized. Mathematically, the problem is similar to completion based on a species tree with the difference being that a set of reference trees (i.e. other gene trees) are available. The benefit of species-tree-free completion is that it may escape the paradoxical reduction in gene tree discordance after completion, and it does not need a reference species tree. To our knowledge, very little work on this question exists. Mirarab (2015) introduced a method for completing gene trees by computing a quartet-based distance matrix from the gene trees and repeated use of the four-point condition. Since this heuristic method was just one step of the larger ASTRAL algorithm, it was not empirically or theoretically evaluated on its own.

Gene tree completion is also intimately connected to another problem: phylogenetic placement given a set of input gene trees. Rabiee and Mirarab (2020b) introduced a method called INSTRAL for adding a new species into a species tree given a set of gene trees that already include the new species while minimizing total quartet discordance between the updated tree and the gene trees. That same mathematical problem can be used to update a gene tree using the other gene trees. When more than one taxon is missing, ordering them in some fashion and repeatedly applying the same algorithm can be used to complete the gene tree. Similarly, Rabiee and Mirarab (2020a) designed a version of ASTRAL that can satisfy constraints, and the constrained version of ASTRAL can be used to complete gene trees.

In this article, we make several contributions. First, we empirically study the species-tree-free gene tree completion problem. While past methods such as INSTRAL can be used to solve the problem, we are not aware of any study that has used them for this purpose. Second, we note that the running time of INSTRAL, which grows quadratically with the size of the backbone tree for each insertion, is sub-optimal. In a seminal work, Brodal *et al.* (2013) introduced an algorithm (called B13 hereafter) that allows computation of the quartet (or triplet) score between two trees in time that grows quasi-linearly with the size of the tree. Here, we extend the B13 algorithm so that it can insert new species into a tree while maximizing its total quartet score with respect to a given set of trees. Thus, we improve the asymptotic complexity of quartet-based taxon insertion (whether for gene trees or species trees). Finally, we introduce some techniques, including subsampling of quartets, that dramatically increase the accuracy of gene tree completion compared with the vanilla application of the optimization problem.

2 Materials and methods

2.1 Notations and definitions

Let $T=(V,E)$ be a single leaf-labeled rooted tree where all edges are directed toward a *root* node denoted by r_T. Let $e=(v,u)$ or e_v for short denote the edge that connects node v to u, and use $u=p(v)$ to denote that u is the *parent* of v and v is a *child* of u. The set of children of an internal node $u\in V$ is denoted as $c(u)$. We give each leaf of T a unique index in the *leafset* $L_T=\{1\dots n\}$. We use $[x]$ as shorthand for $\{0,1,\dots,x\}$. We use n_T and d_T to denote the number of leaves and the maximum degree of any nodes in tree T, respectively (omitting the subscript when clear by context). To *reroot* the tree T at an edge $e_v=(v,u)\in E$, we first divide e into two edges by adding a new vertex r_v to V and replacing e_v with edges (u,r_v) and (v,r_v), then we reverse the direction of all edges on the path from u to r_T, and optionally, remove the old root r_T from V and make each of its children point toward its parent. The resulting graph, T_v, is a new rooted tree with the same topology as T and is called an *alternative rooting* of T on v. We use r_v to denote the root of T_v if we were to reroot T on e_v.

A *triplet* is a subtree of T induced by any three leaves in L_T (note that, because T is single leaf-labeled, a triplet can be uniquely defined by a set of three leaf labels). For each triplet of T, the least common ancestor (LCA) in T of the three leaves is called the *anchor* node of that triplet. A triplet is *resolved* if the LCA of one pair of its species is not the anchor; otherwise, the triplet is *unresolved*. A *quartet* is an *unrooted* subtree of T induced by any four leaves in L_T. Note that, while triplets depend on the rooting of T, quartets do not.

For two trees T_1 and T_2 whose leafsets intersect on a set S of n leaves and a given triplet $\{a,b,c\}\subset S$, we say that T_1 matches T_2 on $\{a,\ b,\ c\}$ if T_1 restricted on $\{a,\ b,\ c\}$ has identical topology to T_2 restricted on $\{a,\ b,\ c\}$. The number of *matching triplets* of T_1 and T_2 is the number of triplets that are shared among $\binom{n}{3}$ triplets on S and is denoted by $\Psi(T_1,T_2)$. For unresolved triplets, we count them as matching only if they are unresolved in both trees. Similarly, for a quartet $\{a,b,c,d\}\subset S$ and two unrooted trees T_1 and T_2, we can define $\Phi(T_1,T_2)$ as the number of quartet topologies that matches between the two trees.

2.2 Problem formulations

We start by defining five interconnected computational problems.

Problem 1 [Maximum-matching quartet placement (MQP)]. Given an unrooted *reference* tree R with $n+1$ leaves and an unrooted *query* tree T on all leaves of R except a leaf x, find an optimal completion T_x of T by placing x onto T to maximize $\Phi(T_x,R)$.

Problem 2 [Maximum-matching triplet rooting (MTR)]. Given a rooted *reference* tree R and an unrooted *query* tree T, find a rooting T_v of T that maximizes $\Psi(T_v,R)$.

Problem 3 [Multi-reference MQP (m-MQP)]. Given a collection of k *reference* trees $\mathcal{R}=\{R_1,R_2,\dots,R_k\}$ all including a leaf x (among other leaves) and a *query* tree T missing x, find a placement T_x of x on T to maximize $\sum_{i=1}^{k}\Phi(T_x,R_i)$.

Problem 4 [Multi-reference MTR (m-MTR).]. Given a collection of k rooted *reference* trees $\mathcal{R}=\{R_1,R_2,\dots,R_k\}$ and an unrooted *query* tree T, find a rooting T_v of T that maximizes $\sum_{i=1}^{k}\Psi(T_v,R_i)$.

The MQP problem is equivalent to the MTR problem: first root R at the taxon x, then remove x from R to obtain R', next solve MTR on R' and T to obtain the optimal rooting of T, and finally, place x on the new root of T to obtain T_x. As proved in the Supplementary Appendix:

Claim 1. *The tree T_x obtained by applying MTR on the query tree T using the reference tree R' and adding x as an outgroup is a solution to the MQP problem on T, x and R.*

Now consider a more general problem:

Problem 5 [Multi-reference multi-query MQP (mm-MQP)]. Given a query tree T *missing a set* X of leaves, and a set of k *reference* trees $\mathcal{R}=\{R_1,R_2,\dots,R_k\}$ with $\cup_1^k L_{R_i}=L_T\cup X$, find a tree T_X on $L_T\cup X$ that is compatible with T and maximizes $\sum_{i=1}^{k}\Phi(T_X,R_i)$.

Note that, the mm-MQP problem is similar to the problem solved by ASTRAL with an input constraint (Rabiee and Mirarab, 2020a), and is NP-hard (Lafond and Scornavacca, 2019). Below, we introduce algorithms to solve MTR, MQP, m-MTR and m-MQP, and a heuristic to solve mm-MQP by sequentially applying m-MQP for each query x in X.

2.3 Algorithms to solve MTR, MQP, m-MTR and m-MQP

We extend the B13 algorithm to compute $\Psi(T_v, R)$ for every rooting T_v of T and select the maximum score. We first assume T and R share the same leafset of size n and then show that it is straightforward to extend the algorithm to trees with different leafsets.

The B13 Overview. To compute $\Psi(T_1, T_2)$, the B13 algorithm traverses T_1 top down, and when a node u is visited, it counts the number of triplets anchoring at u in T_1 that matches T_2. To do so, it colors leaves according to which side of u they belong to. To obtain the quasi-linear complexity, a *Hierarchical Decomposition Tree* (HDT) data structure representing T_2 is maintained. The HDT keeps a set of counters that allow computing the number of matching triplets for the anchor node u of T_1. HDT needs to be updated each time we move to a new node of T_1 and colors change; however, thanks to its careful design that guarantees a locally balanced structure, updating the HDT for each leaf only takes sub-linear time.

Algorithm overview. Naively using the B13 algorithm to examine each edge and choose the one with the maximum score has quasi-quadratic running time. Such a solution would be worse than that of Rabiee and Mirarab (2020b), which is worst-case quadratic time. Here, we extend the B13 algorithm to solve the MTR problem in quasi-linear time (Algorithm 1). When we visit each node u in T in the top-down traversal, we compute several new counters per node (i.e. the number of triplets anchoring at u in T that match the reference tree, the number of triplets anchoring at u in *each alternative rooting* T_{v_i} for the $d-1$ children $v_1 \dots v_{d-1}$ of u that match the reference tree, and the number of triplets anchoring at r_u in T_u that match the reference tree) that allow us to score all possible rootings. To efficiently compute these counters, we also augment the HDT with a new set of counters. Next, we first describe the node coloring scheme, then HDT and its counters, and finally our extensions.

Algorithm 1 Quasi-linear-time algorithm to solve the MTR problem. Color(u, i) colors all the leaves below u with i. d_u is the of degree u.

```
function SolveMTR($T = (V, E), R$)
  HDT rooted at $\mathcal{R} \leftarrow$ Build HDT($R$)    ▷ See Brodal et al. (2013)
  color (root of $T$, 1)
  ColorAndQuery (root of $T$)
  $\Psi$(rootof $T$) $\leftarrow \sum_{u \in \text{internal nodes of } T} \tau^i_u$
  for edge $(v, u)$ in pre-order traversal of $T$ do
    $\Psi(v) = \Psi(u) - \tau^i_u - \tau^r_u + \tau^o_v + \tau^r_v$
  return $T$ rooted at $\text{argmax}_v \Psi(v)$
function ColorAndQuery($u$)
  if $u$ is a leaf then
    color($u$ , 0) and return
  $v_1, v_2, \dots, v_{d_u - 1} \leftarrow c(u)$
  Swap $v_1$ with the largest $v_i$ clade
  for $i = 2$ to $d_u - 1$ do
    color($v_i$, $i$)
  $\pi^{\mathcal{R}}_0, \dots, \pi^{\mathcal{R}}_{d_u - 1}, \rho^{\mathcal{R}} \leftarrow$ update HDT using Equations. (2)–(5)
  $\tau^i_u \leftarrow \pi^{\mathcal{R}}_0$
  $\tau^r_u \leftarrow \rho^{\mathcal{R}}$
  for $i = 1$ to $d_u - 1$ do
    $\tau^o_{v_i} \leftarrow \pi^{\mathcal{R}}_i$
  for $i = 2$ to $d_u - 1$ do
    color($v_i$, 0)
  ColorAndQuery($v_1$)
  for $i = 2$ to $d_u - 1$ do
    color($v_i$ , 1)
    ColorAndQuery($v_i$)
```

2.3.1 Coloring and scoring the query tree T

Consider an arbitrary node u in T (except the root) that has degree d, $p(u) = v_0$, and $c(u) = \{v_1, \dots, v_{d-1}\}$. The node u defines a set of d subtrees on T: the d—1 clades rooted at $v_1, v_2, \dots, v_{d-1}$, and the complement subtree of the clade rooted at u. To *color* T by u, we give all leaves belonging to each subtree of u the same color index $i \in [d-1]$. By convention, the subtree on the direction from u to the root always gets the color 0 (Fig. 1a). When T is colored according to u, each triplet of T that anchors at u must have leaves with two distinct non-zero colors.

Triplet counters of the query tree. To solve the MTR problem, we extend the B13 algorithm to also count the u-anchored triplets of each alternative rooting T_{v_i} of T. These triplets can be determined by the u coloring: each triplet of T_{v_i} anchored at u must have leaves colored with two distinct colors other than i (see Fig. 1b). As the query tree is traversed top down in the B13 algorithm, we update it to compute and store a set of counters for each node v in T (other than the root). Let $u = p(v)$ and recall that r_v is the root of T_v; we maintain the following counters for v:

- τ^i_v: triplets *inside* v. This is the number of triplets anchored at v that match the reference tree.
- τ^o_v: triplets *outside* v. This is the number of triplets anchored at u in the alternative rooting T_v that match the reference tree.
- τ^r_v: triplets at the *rerooting* point. This is the number of triplets anchored at r_v in T_v that match the reference tree.

Note τ^o_v of T equals to τ^i_u of T_v (Fig. 1a and b). Below, we show how to compute these counters using new HDT counters updated after each coloring of T.

Score of alternative rooting. After the first top-down traversal, we compute the triplet score of T (original rooting) to R by summing up τ^i_v for all nodes of T. Then, we compute the triplet score for all alternative rooting T_v of T using a second top-down traversal and the following recursive formula:

$$\Psi(v) = \Psi(p(v)) - \tau^i_{p(v)} - \tau^r_{p(v)} + \tau^o_v + \tau^r_v \,. \qquad (1)$$

Here, to move from the parent to a child, we remove matching triplets anchored at the parent or nodes outside it and add those anchored at the child or any node outside it (a triplet may be added and subtracted back).

2.3.2 Building and using the HDT of the reference tree

Building the HDT. We use the linear-time algorithm of Brodal *et al.* (2013) to build the HDT data structure from the reference tree R. Each node of HDT is a combination of multiple nodes in R carefully selected in a way that ensures the HDT tree is *locally balanced*, meaning that each node with m leaves has $O(m)$ height. This local balance property enables efficient query of the number of matching triplets according to a coloring by an internal node of T. Johansen and Holt (2013) refer to the nodes in the HDT as *components*, each of which is classified into one of the three types: C, G or I (see Supplementary Table S1 and Supplementary Fig. S5, or refer to the original text and fig. 2.5 of Johansen and Holt, 2013). We use terms *node* and *component* interchangeably.

Updating HDT counters. To compute the number of matching triplets, each node of HDT keeps a set of counters (Table 1). These counters only depend on the coloring of leaves, and when a leaf changes color, the HDT counters must be updated. Brodal *et al.* (2013) and Johansen and Holt (2013) have derived recursive formula to compute these counters for each component in the HDT given its children; thus, we can update the counters by visiting all the nodes from the leaf that has changed color to the root.

Let T be colored by node u with degree d_u and children of u, $v_1, \dots, v_{d_u - 1}$ by $1, \dots, d_u - 1$. Note that, $d_u \leq d$ and recall that the subtree above u has color 0. In addition to counters defined by B13, we add the following two sets of counters to each component X of HDT.

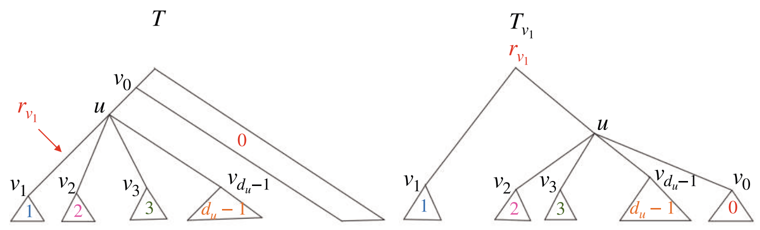

Fig. 1. (**a**) The query tree T colored by node u with degree d_u. Leaves under each v_i are given the same *color* i, and the leaves outside u are colored 0. Any triplet of T anchoring at u must have two leaves taken from leaves under v_i and the other from a clade v_j different from v_i (exclude v_0 as it does not define a clade *below* u). Thus, the colors of a triplet anchoring at u must be (i, i, j) or one of its permutations, where $i \neq j, i, j \in \{1, 2, \dots, d_u - 1\}$. (**b**) The alternative rooting T_{v_1} of T. A new node r_{v_1} is added between u and v_1 to split the edge into two, and the edge directions are adjusted accordingly to have all nodes pointing to the new root. To count the triplets anchoring at u in T_{v_1}, we exclude v_1 instead of v_0 as in T. To count the triplets anchoring at r_{v_1}, we group all colors other than 1 into one group, and count the triplets that have colors $(i, i, 1)$ or $(1, 1, i)$, or a permutation of these two, where $i \neq 1$

- ρ^X: the number of triplets of R that belong to component X and match the corresponding triplets of T_u that are anchored at r_u.
- π_j^X: the number of triplets of R that belong to component X and match the corresponding triplets of T_{v_j} that are anchored at u. If $d_u < d$, we set $\pi_j^X = 0$ for all $j > d_u - 1$.

We now show recursions for ρ^X and π_j^X and prove them in Supplementary Appendix. If X is an I or L, we simply skip it. If X is an IG →C, we copy over the counters of its G child.

If X is a CC →C component with children C_1 and C_2 (by the convention, C_1 is below C_2; see Supplementary Fig. S5), then

$$\pi_j^X = \pi_j^{C_1} + \pi_j^{C_2} + \sum\nolimits_{i \neq j} \left(\binom{n_i^{C_1}}{2} (n_\bullet^{C_2} - n_j^{C_2} - n_i^{C_2}) + n_i^{C_1}(n_{i\uparrow\bullet}^{C_2} - n_{i\uparrow j}^{C_2}) + (n_\bullet^{C_1} - n_j^{C_1} - n_i^{C_1}) n_{(ii)}^{C_2} + n_i^{C_1}(n_{\bullet\square}^{C_2} - n_{j\bullet}^{C_2} - n_{i\bullet}^{C_2} + n_{ij}^{C_2}) \right) \tag{2}$$

$$\rho^X = \rho^{C1} + \rho^{C2} + n_0^{C1} n_{0\uparrow\bullet}^{C2} + (n_\bullet^{C_1} - n_0^{C_1}) \sum_{i=1}^{d} n_{i\uparrow 0}^{C_2} + \binom{n_0^{C1}}{2} (n_\bullet^{C_2} - n_0^{C2}) + \binom{n_\bullet^{C1} - n_0^{C_1}}{2} n_0^{C_2} + (n_\bullet^{C_1} - n_0^{C_1}) n_{(00)}^{C_2} + n_0^{C_1}(n_{(\bullet\square)}^{C_2} - n_{(0\bullet)}^{C_2}) \tag{3}$$

If X is a GG →G component with children G_1 and G_2, then

$$\pi_j^X = \pi_j^{G_1} + \pi_j^{G_2} + \sum\nolimits_{i \neq j} (n_i^{G_1}(n_{\bullet\square}^{G_2} - n_{j\bullet}^{G_2} - n_{i\bullet}^{G_2} + n_{ji}^{G_2}) + n_i^{G_2}(n_{\bullet\square}^{G_1} - n_{j\bullet}^{G_1} - n_{i\bullet}^{G_1} + n_{ji}^{G_1}) + n_{(ii)}^{G_1}(n_\bullet^{G_2} - n_j^{G_2} - n_i^{G_2}) + n_{(ii)}^{G_2}(n_\bullet^{G_1} - n_j^{G_1} - n_i^{G_1})) \tag{4}$$

$$\rho^X = \rho^{G1} + \rho^{G2} + n_{00}^{G_1}(n_\bullet^{G_2} - n_0^{G_2}) + (n_{\bullet\square}^{G_1} - n_{(0\bullet)}^{G_1}) n_0^{G_2} + n_{00}^{G_2}(n_\bullet^{G_1} - n_0^{G_1}) + (n_{\bullet\square}^{G_2} - n_{(0\bullet)}^{G_2}) n_0^{G_1} \tag{5}$$

These HDT counters readily give us the $d + 1$ counters associated to node u of T as defined earlier. More precisely, $\tau_u^i = \pi_0^{\mathcal{R}}$, $\tau_u^r = \rho^{\mathcal{R}}$ and $\tau_{v_j}^o = \pi_j^{\mathcal{R}}$ for each $j = [d_u]$ where $\mathcal{R}$ is the root of the HDT.

2.3.3 Generalizations

Note that, Algorithm 1 computes and stores the score $\Psi(T_v, R)$ for *every* alternative rooting T_v of T. Thus, solving m-MTR is straightforward: we first apply Algorithm 1 to each reference tree R_i and T to compute $\Psi(T_v, R_i)$ for *all* node v of T. Then, for each node v of T, we compute $\Psi_v = \sum_{i=1}^{k} \Psi(T_v, R_i)$. Finally, we select the node v^* with maximum Ψ_{v^*} and reroot T at v^*. By Claim 1, this algorithm also solves m-MQP.

Algorithm 1 can be adopted to cases where T and R have different leafsets $L_T \cap L_R = S$ with minor modifications. Because the leaves in $L_R \setminus S$ and $L_T \setminus S$ do not contribute to the number of matching triplets, we can simply ignore them. To do so, we restrict R on S by removing from R all the leaves in $L_R \setminus S$. We mark all the leaves in T that are not in S as *inactive* and ignore the inactive leaves by not coloring them during the top-down traversal of T. The resulting algorithm is clearly correct.

2.4 Complexity analysis

Thanks to the *smaller-half* trick of Brodal *et al.* (2013), at most $O(n \log n)$ leaves change color in the (recursive) top-down traversal of Algorithm 1 (i.e. the ColorAndQuery function). Therefore, the coloring module performs at most $O(n \log n)$ operations. To incorporate our extensions, three extra counters are maintained for each node in T, all of which are computed in $O(1)$ using the same top-down traversal for coloring. Thus, the asymptotic complexity of coloring does not change. The B13 algorithm builds HDT in linear time. Because the HDT has $O(n)$ components and is locally balanced, the original HDT used in B13 can be queried in $O(\log n)$ per leaf recoloring (see Brodal *et al.* (2013) and Johansen and Holt (2013)). Our extensions require $O(d^2)$ counters per HDT component (instead of $O(1)$ counters used in B13), so the complexity per HDT query increases by a factor of $O(d^2)$. Thus, the total time complexity of Algorithm 1 is $O(d^2 n \log^2 n)$. In a tree where d is bounded by a constant (e.g. a fully resolved binary tree), the complexity is $O(n \log^2 n)$. With k reference trees, the time complexity of both m-MTR and m-MQP is $O(kd^2 n \log^2 n)$.

2.5 tripVote: completing gene trees (mm-MQP)

We develop a heuristic method using m-MQP to complete a set of incomplete gene trees (mm-MQP). To complete a gene tree T_i, we sequentially apply the m-MQP algorithm to place each missing taxon onto T_i, using the other gene trees as references. This greedy algorithm optimizes the number of shared quartets with reference trees that include at least three of their four leaves coming from T_i. However, it does not solve the NP-Hard problem (Lafond and Scornavacca, 2019) of finding a complete tree with optimal quartet score over all quartets. Thus, the order of placements can change the outcome. In tripVote, we order missing taxa by their descending frequency of presence in the input gene trees, breaking ties arbitrarily. Note that tripVote only works on single-labeled gene trees.

2.5.1 Quartet sampling

As long appreciated, quartets with shorter terminal branches (i.e. short quartets) have better theoretical (Erdos *et al.*, 1999) and empirical (Snir *et al.*, 2008) performance than long quartets, motivating some quartet-based methods to select short quartets (e.g. Nelesen *et al.*, 2012; Warnow *et al.*, 2001). Inspired by these methods, we propose a stochastic approach to down-weight the votes of long quartets around the query taxon in reference trees. After rooting a gene tree at the query taxon, we sample random paths from the root to a leaf, selecting a child of a node uniformly at random at each step (Supplementary Fig. S4). For each reference tree, we sample s taxa with replacement, then remove duplicates and restrict the tree to the selected set of taxa. We repeat this sampling procedure r times

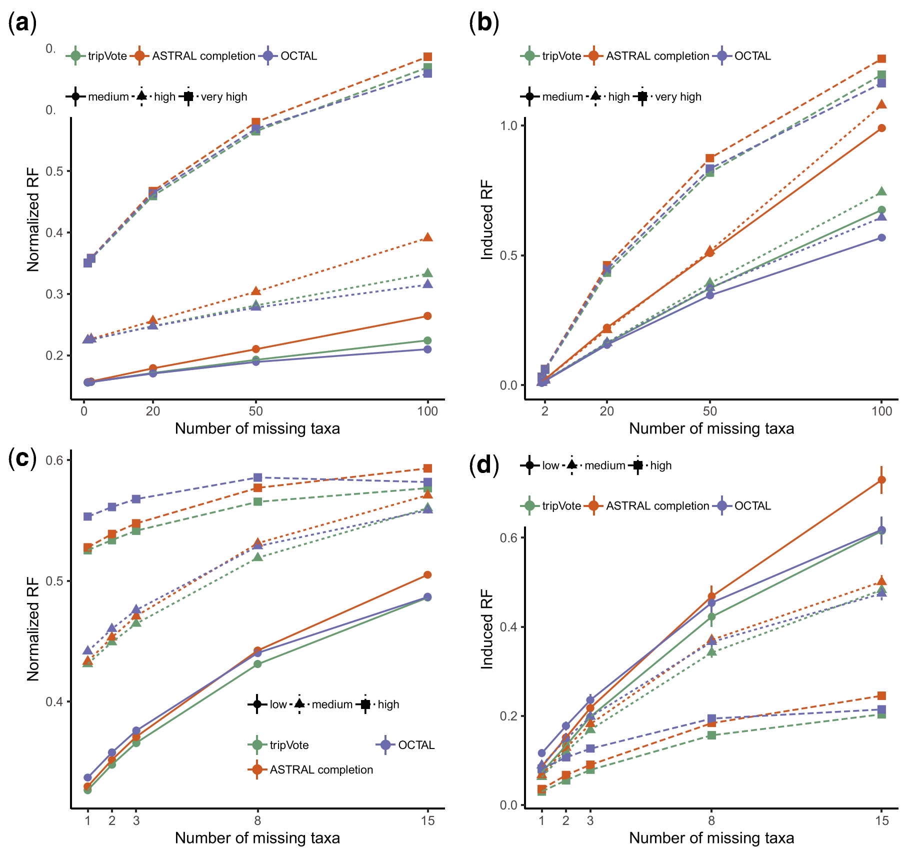

Fig. 2. (**a** and **c**) Normalized RF error of tripVote, OCTAL and ASTRAL-completion on the 201-taxon dataset with different levels of ILS (a), and the 31-taxon dataset with different levels of clock deviations (c); $m = 0$ shows the average RF error of the complete gene trees estimated by FastTree. (**b** and **d**) Induced RF error of tripVote, OCTAL and ASTRAL-completion on the 201-taxon dataset (**b**) and the 31-taxon dataset (**d**). See Supplementary Figures S8 and S9 for the random completion as control

Table 1. HDT counters

Entity	Is the number of ...
n^X_i	*i*-colored leaves below *X*.
n^X_{ij}	Pairs of leaves below *X* where one is colored *i*, the other is colored *j*. If *X* is a C type, the LCA of these two leaves must be on the external path of *X* and the path from the LCA to either of these two leaves does not pass through any other node on the external path. If *X* is a G type, the two leaves must belong to two distinct subtrees of the super root of *X* (Supplementary Fig. S5).
$n^X_{i\uparrow j}$	Pairs of leaves where one is colored *i*, the other is colored *j* and the *i*-colored leaf is below the *j*-colored leaf in *X* (only defined for C type; see Supplementary Fig. S5).
$n^X_{(ii)}$	Pairs of leaves below *X* both with color *i*. If *X* is a C type, the LCA of these leaves must not belong to the external path. If *X* is a G type, the LCA must not be the super root of *X*.
$n^X_{(0\bullet)}$	Pairs of leaves below *X* with one leaf colored 0 and the other colored different from 0 whose LCA is *not* a node on the external path if *X* is a C component or the super root if *X* is a G component.

Note: Everywhere, $i, j \in [d]$. As in Johansen and Holt (2013), we use the descriptors • and □ to represent any color (unlike Johansen and Holt (2013), we include 0, which is needed for rooting). Thus, $n^X_{i\uparrow\bullet} = \sum_{j\in[d]} n^X_{i\uparrow j}$; $n^X_{i\bullet} = \sum_{j\neq i} n^X_{ij}$; $n^X_{\bullet} = \sum_i n^X_i$; $n^X_{\bullet\square} = \sum_i n^X_{i\bullet}$

to generate *r* sampled trees for each reference tree. After sampling, we combine all the generated sample trees across all genes as the reference trees in *m*-MQP.

While hyperparameters *s* and *r* can be set by users, by default, we choose them such that leaves close to the root have a sufficiently high probability of being sampled at least once across the $s \times r$ draws. We first set *s* such that a taxon at the distance $\frac{1}{2}\log_2 n$ from the root is expected to be sampled once in each round. Since the probability of sampling a leaf at distance $\frac{1}{2}\log_2 n$ from the root in one traversal is $\frac{1}{2^{(\log_2 n)/2}} = \frac{1}{\sqrt{n}}$, setting $s = \sqrt{n}$ achieves the goal. Thus, we choose $s = \lceil\sqrt{n}\rceil$. To select the number of rounds, we set *r* such that a taxon with distance at most *h* (a predefined constant) from the root is sampled with high probability. That is, for a false-negative tolerance ϵ, we require: $1 - (1 - \frac{1}{2^h})^{sr} > 1 - \epsilon$. By default, we set $h = 5$ and $\epsilon = 0.05$; thus, $s \times r = \frac{\log(0.05)}{\log(31/32)} \approx 95$ to satisfy the above inequality. Thus, by default $s = \lceil\sqrt{n}\rceil$ and $r = \frac{95}{\lceil\sqrt{n}\rceil}$.

2.5.2 Software package

We updated the C++ software by Sand *et al.* (2014) to incorporate our algorithm to solve MTR. We built a Python wrapper, tripVote, for the C++ package and added new functions for gene tree

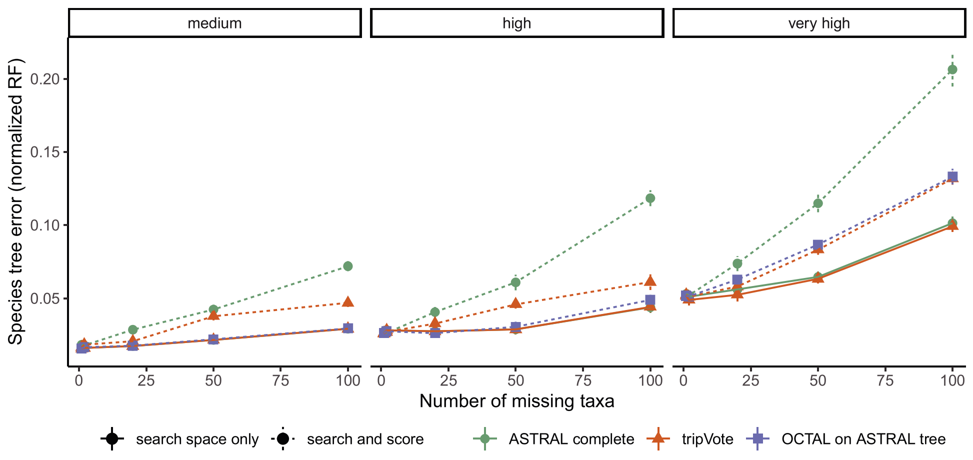

Fig. 3. Topological error of the ASTRAL species tree estimated using different set of gene trees (the 201-taxon dataset). The three panels show different levels of ILS. In 'search space only', the completed gene trees (by ASTRAL-complete, tripVote or OCTAL) were only used to guide ASTRAL's search space, whereas in 'search and score', the completed gene trees were used as the actual input to ASTRAL. To obtain the results for OCTAL, two rounds of ASTRAL was run: in the first round the search space was produced by ASTRAL-complete given the incomplete trees; in the second round, ASTRAL was run using the OCTAL-completed gene trees, both for search space and as input

completion using mm-MQP heuristics, with or without the quartet sampling strategy.

3 Evaluation procedures

3.1 Datasets

We test tripVote on published simulated datasets by Mirarab and Warnow (2015) and Mai *et al.* (2017) and a real plant dataset by OneKP Initiative (2019). The simulated datasets were both created using Simphy to generate gene and species trees under the multi-species coalescent model and heterogeneous parameters, and Indelible to simulate nucleotide sequences on gene trees according to the GTR + Γ model with varying sequence lengths and sequence evolution parameters. FastTree2 was used to estimate gene trees based on the GTR + Γ model. Original papers provide full details on the parameters used in each of these two datasets.

The **201-taxon** dataset by Mirarab and Warnow (2015) enables us to examine the effect of incomplete lineage sorting (ILS) on gene tree completion methods. From this dataset, we use 3 model conditions with 200 taxa where tree length is 2M and speciation rate is 1e-6, and use the first 20 out of the 50 replicates of the original dataset (to save computational time). In each replicate, we use the first 500 (out of 1000) estimated gene trees that are fully resolved. The three model conditions have medium, high and very high levels of ILS, resulting in 21, 33 and 69% RF distance between true gene trees and the species tree. They also have high levels of gene tree estimation error (15, 22 and 34% RF between estimated and true gene trees).

The **31-taxon** dataset by Mai *et al.* (2017) is used to examine the effect of clock deviations on gene tree completion methods. Here, we use the three model conditions with the root to crown ratio equal to 1.0, but varying levels of deviation from the clock (low, medium, high). We only use the first 20 out of the 100 replicates of the original dataset because our experiments are computationally intensive. The average coefficient of variation of root-to-tip distances of low, medium and high deviations are 0.04, 0.30 and 0.69, respectively. These replicates have moderately high level of ILS (with 24% mean RF distance between true gene trees and the species tree). The amount of gene tree estimation error increases with deviations from the clock (RF errors are 30, 41, and 52%).

The real OneKP biological dataset of 1178 plants by OneKP Initiative (2019) has 384 gene trees, all of which miss some of the species. The original study provide an ASTRAL species tree inferred from 384 gene trees, inferred using RAxML, each with at least $1178/2 = 589$ species.

3.2 Experiments

We compare tripVote with two alternatives tree completion algorithms: *ASTRAL-completion*, the method used in ASTRAL and described in Mirarab (2015), and *OCTAL*, the gene tree completion method that minimizes RF distance of each gene tree to the species tree. ASTRAL-completion is run using the ASTRAL software, and OCTAL is run using the TRACTION-RF software (Christensen *et al.*, 2018). In addition, to guide visualization and interpretation, we add a lower-bound control by randomly completing the gene trees (repeated 100 times and averaged).

In simulated datasets, we randomly remove m leaves from each estimated gene tree to create incomplete gene trees; $m \in \{0, 1, 2, 20, 50, 100\}$ for the 201-taxon dataset and $m \in \{0, 1, 2, 3, 8, 15\}$ for the 31-taxon dataset. We use tripVote, OCTAL and ASTRAL-completion to complete each set. For the 201-taxon dataset with $m = 1$, we compare the accuracy of tripVote with and without the sampling.

We use two different error metric: the normalized *RF* distance and the *induced RF distance*, as described below. To separate the gene tree estimation error from the error by completion methods, we define the *induced* (normalized) RF distance, as follow: given two trees T_1, T_2 and a reference tree R, the induced RF distance of T_2 on T_1 with respect to R is $\frac{RF(T_2,R)-RF(T_1,R)}{RF(T_1,R)}$ where RF denotes the normalized RF distance of two trees after restricting them to their shared leafset. Positive (negative) induced RF distances show that T_1 (T_2) is closer to the reference tree. On the simulated datasets, we use the estimated gene tree by FastTree as T_1, the tree completed by a completion method (e.g. ASTRAL-completion, tripVote, etc.) as T_2, and the true gene tree as R.

In addition, we test the ability of tripVote to improve species tree estimation. On the 201-taxon dataset, we compare five versions of ASTRAL for inferring species trees from incomplete gene trees. (i) The default ASTRAL uses ASTRAL-completion to construct the search space and original trees to score. (ii) We use tripVote in place of ASTRAL-completion but continue to score trees using incomplete trees. (iii) We use OCTAL in place of ASTRAL-completion. Since running OCTAL needs a species tree, we use the ASTRAL species tree inferred in (i) as input to OCTAL. Thus, in this setting, ASTRAL is run twice. (iv) We use the gene trees completed by ASTRAL-completion as input to ASTRAL, making them used both for search space creation and scoring. (v) Similarly, we use tripVote completed trees as input. We measure the error of these ASTRAL trees by computing their RF distances to the true species tree.

We also test tripVote and ASTRAL-completion on their ability to root an unrooted gene tree with respect to other rooted gene trees. On the two simulated datasets, we remove the outgroup from a set of $n - k$ gene trees (arbitrarily selected) and use the k remaining trees to infer the outgroup placement. We vary k in $\{1, 10, 50, 100, 250, 500\}$. To measure error, we compute the optimal rooting that minimizes the triplet distance to the true tree and report the *delta triplet error*, defined as the difference between the triplet distances of a rooted tree and the optimal tree to the true tree. In addition to ASTRAL-completion, we also compare tripVote to other rooting methods, including the outgroup rooting (root at the original placement of the outgroup before removing it), mid-point rooting and MinVar rooting (Mai *et al.*, 2017) and add the random rooting as a control.

For the OneKP dataset, we set up two versions: one where the *original* gene trees are used directly and one with *extra missing data* where we prune out an extra p% of the taxa from each gene tree (for $p \in \{5, 10, 15, 20\}$). With original data, where the completed gene trees are unknown, we measure the induced RF distance of the completed gene tree (T_2) on the original (incomplete) one (T_1) with respect to the species tree (R). For the extra missing data, after running the methods to complete the gene trees, we reduce the completed gene trees to the same leafset as the original gene trees and compute their normalized RF distances.

4 Results

4.1 Simulated datasets

4.1.1 Gene tree completion

Across all model conditions with $m = 1$, the sub-sampling strategy dramatically lowers the error compared to full quartet sampling (Supplementary Fig. S6). Both versions of tripVote have higher error when the ILS level increases. The averaged error of tripVote with and without sampling are 0.51 versus 0.84, 0.77 versus 1.75, and 3.77 versus 5.42 for medium, high and very high ILS, respectively. Thus, the error is less than half for the high ILS level and is reduced everywhere. Looking beyond the average error and examining the full distribution shows that while in the majority of cases error is at most one branch with sampling, the same is not true when sampling is not performed. Both versions suffer from a tail of placements with very high error (e.g. five edges or more), a condition that unsurprisingly is observed mostly for the highest level of ILS. However, the tail of large errors is clearly reduced after sampling. Because restricting the calculations to shorter quartets has a clear positive impact on the results, we use sampling by default in tripVote and use it in the remaining experiments.

Comparing tripVote and ASTRAL-completion, across all conditions, tripVote always has lower error and the difference between the two methods is more pronounced when the number of missing taxa increases (Fig. 2). The relative improvements of tripVote compared with ASTRAL-completion are quite large. On the 201-taxon dataset at 50% missing data (i.e. $m = 100$), the induced RF error of tripVote is 32%, 34%, and 6% lower than that of ASTRAL-completion in medium, high and very high ILS levels, respectively (Fig. 2b). Similarly, tripVote dominates ASTRAL-completion on the 31-taxon dataset across all conditions of clock deviation, albeit with smaller differences compared to 201-taxon dataset. For example, with $m = 15$, the induced RF error of tripVote is 11%, 2%, and 4% lower in low, medium and high clock deviations, respectively (Fig. 2d).

The comparison between tripVote and OCTAL depends on the dataset and the level of missing data. On the 31-taxon dataset, tripVote has better accuracy, and the improvements are most pronounced with higher clock deviations and medium level of missing data (e.g. eight taxa). On the other hand, OCTAL is more accurate in most conditions of the 201-taxon dataset and especially when the amount of missing data exceeds 50 taxa. Improvements of OCTAL over tripVote are non-existent or negligible for the highest levels of ILS and are increased for lower levels.

All methods are affected by the level of missing data, ILS (Fig. 2a and b) and clock deviations (Fig. 2c and d). Even before completion, gene trees have higher levels of errors when the ILS is higher or when the deviations from the clock are more pronounced.

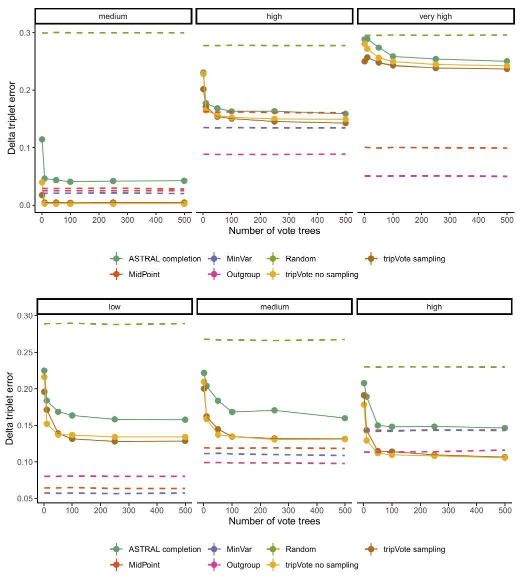

Fig. 4. Accuracy of rooting based on different methods for (**a**) The 201-taxon dataset and (**b**) The 31-taxon dataset. The outgroup is removed from m randomly selected trees and inserted back using either ASTRAL-completion or tripVote, then each of these trees is rerooted at the reinserted outgroup. The x-axis shows the number of voting trees for ASTRAL-completion and tripVote (i.e. $n - m$) and the y-axis shows the delta triplet error (i.e. the triplet error to the true rooted tree subtracting the triplet error of the optimal rooting that has minimum triplet error to the true tree). We added alternative rooting methods (Outgroup, MinVar, MidPoint and Random) that do not use other gene trees (dotted lines). Outgroup rooting was done on the complete estimated trees with outgroup included. MidPoint and MinVar were run *after* the outgroup was removed. The Random rooting was repeated 50 times and the average error is reported. See also Supplementary Figure S3 where the error is measured by the raw triplet error

Completion always increases error compared with estimated gene trees, especially when there are more missing data. However, this increase in error is more pronounced for the highest level of ILS than lower levels. Thus, for very high ILS, not only gene tree estimation is difficult, completion is also difficult. In particular, RF distances after completion can reach 0.7 for the highest ILS case. In contrast, average levels of RF remain below 0.33 after completion for medium or high ILS. Thus, gene tree-based completion using tripVote fails to be accurate at the highest levels of ILS. In contrast to ILS levels, we did not detect a reduction in effectiveness of tripVote as deviations from the clock increase. In fact, induced RF errors go down with increasing levels of clock deviations (Fig. 2d). Note that, with high deviations, the error is already very high before completion and there is relatively little room left for increased error.

4.1.2 Effects on species tree accuracy

We ran ASTRAL to infer the species trees from incomplete gene trees using five methods described earlier and compared their normalized RF errors (Fig. 3). All ways of running ASTRAL showed *some* level of sensitivity to missing data, especially for high ILS and more than 50 missing taxa per gene (≈25% of the leaves). In contrast, the condition with the lowest level of ILS is remarkably robust to even extreme levels of missing data (≈50% of the leaves).

At all levels of ILS, the accuracy is always higher when the completed gene trees are only used to construct search space than when they are also used for scoring species trees. Overall, the best accuracy is obtained when tripVote is used only for building the search space. In this setting, tripVote slightly improves upon the default ASTRAL-completion method when ILS is very high and there is moderate amount of missing data (i.e. up to 50 taxa). Thus, tripVote can be used in place of ASTRAL-completion inside ASTRAL to improve its accuracy. Moreover, tripVote has far better accuracy than ASTRAL-complete when the completed gene trees are used both for search space and scoring. Comparing with the original setting of ASTRAL (which uses ASTRAL-completion for search space only), the ASTRAL tree inferred using OCTAL either has the same accuracy (when ILS is medium) or worse. Note that, the OCTAL setting uses the default ASTRAL species as input. Therefore, our results do not show any benefit in using OCTAL for improving the search space of ASTRAL.

4.1.3 Gene tree rooting

The accuracy of tripVote for rooting is mixed. The absolute error of tripVote rooting clearly increases with the level of ILS (Fig. 4, top), but not with deviations from the clock (Fig. 4, bottom). The accuracy of tripVote (and also ASTRAL-completion) rapidly increases as the number of voting trees increases to 100, but there is relatively little improvement after that. Overall, the accuracy of tripVote improves as a result of adding the sampling strategy; however, the improvements are more subtle than those observed for the placement problem.

In all model conditions, tripVote is more accurate than ASTRAL-completion, but its accuracy comparing to other methods depends on the model condition. With medium ILS, tripVote is the best method and even outperforms outgroup rooting (Fig. 4a). With high ILS, tripVote is similar in accuracy to MinVar. However, when ILS is very high, tripVote is not a good choice (Fig. 4a). Overall, if an outgroup is available, it is clearly a better choice than tripVote when ILS levels are high or very high. When clock deviation is low, branch-length-based rooting methods are very accurate and better than outgroups and tripVote. (Fig. 4b). In medium clock, the error of MinVar and MidPoint go up but still slightly dominate tripVote, and outgroup rooting is the most accurate. When the clock deviation is high, MinVar and MidPoint have higher error, and tripVote is the best method given enough number of voting trees (Fig. 4b).

4.1.4 Running time

We note that tripVote, if run without the sampling strategy to complete the species tree, solves a similar problem to INSTRAL. Using the dataset from the original study by Rabiee and Mirarab (2020b), we compare the running time of INSTRAL and the two versions of tripVote with and without sampling (Supplementary Fig. S2). Here, the species tree (not a gene tree) is being completed, and the two methods are guaranteed to return the same solution; the only difference is the running time. The running times of the two methods are comparable when they both use complete input gene trees as input. The theoretical running time of INSTRAL depends on the number of *unique* nodes across all gene trees (e.g. tripartitions for a binary tree), and thus, very similar gene trees do not increase its running time dramatically. However, in practice, gene trees often miss at least *some* leaves, forcing most nodes to be distinct. Thus, we also tested a case where gene trees missed only 1% of the leaves. Under these conditions, tripVote is much faster. For example, with 10 000 species, INSTRAL takes on average 71 min while tripVote takes only 14 min.

Consistent with the theory, the asymptotic running time of INSTRAL grows faster than linearly (close to $n^{1.4}$) without missing data and close to quadratically with missing data. In contrast, tripVote running time without sampling increases close to linearly with or without missing data. With sampling, because we set the sampling size to a sublinear function of n, the running time of tripVote further reduces and is sublinear (close to $n^{0.9}$).

4.2 Real datasets

On the real dataset, we show the incongruence of the completed gene trees (original data) with the species tree (Fig. 5a), the error of the completed gene trees at different levels of extra missing data (Fig. 5b), and the estimated branch length of the species tree (for original and extra missing data at 20%, Fig. 5c). Consistent with the results of simulated data, here we also see that tripVote is more accurate than ASTRAL-complete, but is not as accurate as OCTAL, especially with higher levels of missing data. Thus, in terms of topological accuracy of gene trees alone, using the species tree to complete the gene trees gives the best results.

The completed OCTAL trees, however, are biased. Ideally, the completed trees should be no more or less distant to the species tree than the original incomplete trees, and the induced RF distance should be distributed around 0. Both the random and the OCTAL methods substantially change the distance to the species tree, especially when the number of missing taxa increases. The random completion sharply increases the induced RF distance with a high variance. While the induced RF distance of the OCTAL method has very low variance at all levels of missing data, the value decreases below 0 when the missing data increases. This reduction shows that the OCTAL method produces completed gene trees that have *lower* discordance with the species tree than incomplete gene trees, and indicates that the resulting completed trees may be overfit to the species tree. ASTRAL-completion and tripVote have relatively little effect on induced RF distance and keep it around 0 even at the highest levels of missing data. The two methods have the opposite tendencies: ASTRAL-completion tends to slightly increase the distance to species tree (mean induced RF: 0.035) while tripVote tends to slightly decrease the distance (mean induced RF: −0.015). Also, ASTRAL-completion has a higher variance compared to tripVote (0.006 versus 0.002).

As a result of these biases, when OCTAL-completed gene trees are used to estimate the species tree, the OCTAL trees cause an overestimation in the species tree branch lengths compared with using the original gene trees (Fig. 5c). Such a problem is far less severe when tripVote or ASTRAL-completion is used. Both original data and the extra 20% missing data show a consistent pattern. As

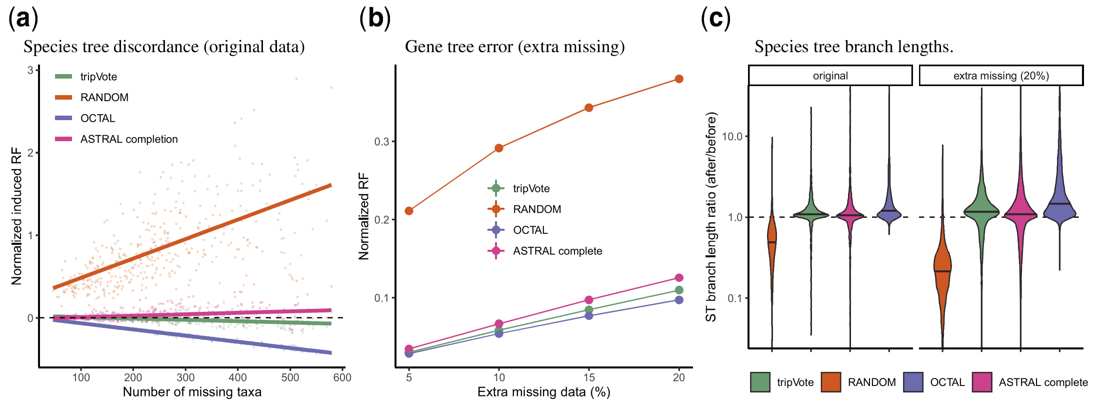

Fig. 5. The OneKP results for (**a** and left panel of c) completing incomplete gene trees, and (**b** and right panel of c) completing gene trees with extra (introduced) missing data. (**a**) Induced RF distance to the species tree of different completion methods on the original incomplete gene trees versus the number of missing taxa. (**b**) The ratio of the species tree branch lengths after versus before completion by different methods; the *y*-axis is shown in logarithmic scale. See Supplementary Figure S7 for normalized RF and another view of the branch length estimation

expected, the branches of the species tree estimated using random completed gene trees are underestimated compared with the original branch lengths obtained from incomplete gene trees.

5 Discussion

We introduced a quasi-linear time algorithm for adding a new taxon to a tree to maximize its total matching quartets to a given set of reference trees that already include the taxon. We built a method called tripVote around this algorithm using a sampling strategy to further improve accuracy and a simple greedy algorithm to allow adding multiple taxa. Overall, results indicate that species-tree-free completion of gene trees does add to the error of the trees, compared with what could be achieved if sequences were available. This much should not be surprising. Gene-tree-based completion was also not always more accurate than species tree aware completion. However, results indicate that gene-tree-based completion is able to maintain the overall levels of gene tree discordance with the species tree. Thus, unlike species tree aware completion, the method does not seem biased toward increasing or decreasing the gene tree discordance. Two main factors limiting the accuracy of gene tree completion seem to be the true levels of gene tree discordance (e.g. ILS) and the amount of gene tree error (controlled in our experiments using deviations from the clock).

Comparing the species tree aware method, OCTAL, with tripVote, we saw mixed results. While tripVote has better accuracy with higher clock deviations and moderate levels of missing data, OCTAL is more accurate in other settings and the gap increases with the number of missing taxa. While part of the differences may be due to the inherent advantage of using a species tree, the more subtle issue of optimality needs to be also considered. While OCTAL is an exact algorithms that minimizes the RF distance of each gene tree to the species tree, tripVote is a greedy heuristic when there are more than one missing taxon. Its heuristic nature may explain why tripVote's accuracy degrades with the level of missing data more quickly than OCTAL, as its error after each m-MQP application can add up. Note that, since OCTAL requires a species tree to operate, it has two limitations: it makes the completed gene trees biased toward the species tree used and it is not useful for the species tree estimation problem (even in the two-iteration setting where we tested it). In contrast, tripVote works directly on a set of gene trees and maintains a more faithful distribution of the gene trees discordance after completion. Therefore, tripVote is more suitable than OCTAL in use cases that need to maintain the discordance level or have to avoid the use of a reference tree.

While we tested tripVote for gene tree completion, the MQP and *m*-MQP algorithms can be used in other contexts such as species tree completion. In that usage, tripVote (without sampling) would be identical to INSTRAL in terms of the resulting placement (both solve the same problem exactly) but will have a better worst-case running time complexity. This better running time also opens the door for developing methods that can infer an entire species tree by repeated placements (i.e. using a greedy algorithm to solve an NP-Hard problem). While a simple greedy search may not outperform methods such as ASTRAL (Zhang *et al.*, 2018), repeated applications of the greedy search may provide a better running time versus optimality tradeoff. We leave the exploration of such directions to future work.

The ability of tripVote to root trees was mixed and depended on the dataset. Given the difficulty of knowing the model condition on real data, we do not necessarily advocate using tripVote for rooting, unless when researchers know the levels of ILS are not high and *some* deviations from the clock are expected. Otherwise, using methods such as MinVar seems preferable. Future works can improve the rooting accuracy by combining tripVote and branch-length-based rooting. One direction could be incorporating the MinVar score of each branch in addition to the tripVote score, but that approach requires a way to combine the two scores. Taking the idea further, machine learning techniques could perhaps be used to combine the scores from multiple methods to find the best rooting overall by training for parameters of a function that combines these scores as features.

The tripVote method can also be improved in several ways. First, since tripVote is a greedy algorithm, the ordering of the taxa to be inserted may affect its accuracy. Future works can explore different strategies to order the queries or run multiple times and summarize results across multiple orderings. Second, the current setting gives the same weight to the vote of every reference tree, regardless of its distance to the backbone tree. As the topology of different gene trees can vary substantially, a weighting scheme that discounts the votes of distant gene trees to the backbone tree should be explored. Finally, while tripVote computes all the individual votes of every reference tree, it only uses their sum to select the placement branch. Another research direction is to explore other strategies to summarize the votes, such as using the median, or a non-linear transformation of each triplet score before summing. Alternatively, one can also take a machine learning approach to use the set of votes from the reference trees as features to learn and predict the best placement branch, in a framework such as that of Jiang *et al.* (2021).

Funding

This work was supported by the National Science Foundation (NSF) [IIS-1845967 to U.M. and M.R.]. Computations were performed on the San Diego Supercomputer Center (SDSC) through XSEDE allocations, which was supported by the NSF [ACI-1053575].

Conflict of Interest: none declared.

Data availability

All data used in this paper and the tripVote software are available at https://uym2.github.io/tripVote/

References

Aiemvaravutigul,C. and Wongwattanakij,N. (2019) A linear-time algorithm for optimal tree completion. In: *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, Chonburi, Thailand.

Bansal,M.S. (2018) Linear-time algorithms for some phylogenetic tree completion problems under robinson-foulds distance. In: *RECOMB International conference on Comparative Genomics*. Springer, Paris, France, pp. 209–226.

Bayzid,M.S. and Warnow,T. (2012) Estimating optimal species trees from incomplete gene trees under deep coalescence. *J. Comput. Biol.*, **19**, 591–605.

Brodal,G.S. *et al.* (2013) Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13. Society for Industrial and Applied Mathematics, USA, pp. 1814–1832.

Christensen,S. *et al.* (2018) Octal: optimal completion of gene trees in polynomial time. *Algorithms Mol. Biol.*, **13**, 6.

Doyle,V.P. *et al.* (2015) Can we identify genes with increased phylogenetic reliability? *Syst. Biol.*, **64**, 824–837.

Erdos,P. *et al.* (1999) A few logs suffice to build (almost) all trees: part II. *Theor. Comput. Sci.*, **221**, 77–118.

Hosner,P.A. *et al.* (2016) Avoiding missing data biases in phylogenomic inference: an empirical study in the Landfowl (Aves: Galliformes). *Mol. Biol. Evol.*, **33**, 1110–1125.

Hovmöller,R. *et al.* (2013) Effects of missing data on species tree estimation under the coalescent. *Mol. Phylogenet. Evol.*, **69**, 1057–1062.

Jiang,Y. *et al.* (2021) DEPP: deep learning enables extending species trees using single genes. *bioRxiv (abstract in RECOMB 2021)*, Padova, Italy, 2021.01.22.427808.

Johansen,J. and Holt,M.K. (2013) *Computing Triplet and Quartet Distances.* Master's thesis, Department of Computer Science, Aarhus University, Aarhus, Denmark.

Lafond,M. and Scornavacca,C. (2019) On the weighted quartet consensus problem. *Theor. Comput. Sci.*, **769**, 1–17.

Laurin-Lemay,S. *et al.* (2012) Origin of land plants revisited in the light of sequence contamination and missing data. *Curr. Biol.*, **22**, R593–R594.

Mai,U. and Mirarab,S. (2018) TreeShrink: fast and accurate detection of outlier long branches in collections of phylogenetic trees. *BMC Genomics*, **19**, 272.

Mai,U. *et al.* (2017) Minimum variance rooting of phylogenetic trees and implications for species tree reconstruction. *PLoS One*, **12**, e0182238.

Mclean,B.S. *et al.* (2019) Impacts of inference method and data set filtering on phylogenomic resolution in a rapid radiation of ground squirrels (Xerinae: Marmotini). *Syst. Biol.*, **68**, 298–316.

Mirarab,S. (2015) Novel scalable approaches for multiple sequence alignment and phylogenomic reconstruction. http://hdl.handle.net/2152/31377.

Mirarab,S. and Warnow,T. (2015) ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics*, **31**, i44–i52.

Molloy,E.K. and Warnow,T. (2018) To include or not to include: the impact of gene filtering on species tree estimation methods. *System. Biol.*, **67**, 285–303.

Nelesen,S.M. *et al.* (2012) DACTAL: divide-and-conquer trees (almost) without alignments. *Bioinformatics*, **28**, i274–i282.

Nute,M. *et al.* (2018) The performance of coalescent-based species tree estimation methods under models of missing data. *BMC Genomics*, **19**, 133.

OneKP Initiative,O.T.P.T. (2019) One thousand plant transcriptomes and the phylogenomics of green plants. *Nature*, **574**, 679–685.

Philippe,H. *et al.* (2017) Pitfalls in supermatrix phylogenomics. *Eur. J. Taxonomy*, **280**, 1–25.

Rabiee,M. and Mirarab,S. (2020a) Forcing external constraints on tree inference using ASTRAL. *BMC Genomics*, **21**, 218.

Rabiee,M. and Mirarab,S. (2020b) INSTRAL: discordance-aware phylogenetic placement using quartet scores. *Syst. Biol.*, **69**, 384–391.

Robinson,D. and Foulds,L. (1981) Comparison of phylogenetic trees. *Math. Biosci.*, **53**, 131–147.

Sand,A. *et al.* (2014) tqDist: a library for computing the quartet and triplet distances between binary or general trees. *Bioinformatics*, **30**, 2079–2080.

Sayyari,E. *et al.* (2017) Fragmentary gene sequences negatively impact gene tree and species tree reconstruction. *Mol. Biol. Evol.*, **34**, 3279–3291.

Snir,S. *et al.* (2008) Short quartet puzzling: a new quartet-based phylogeny reconstruction algorithm. *J. Comput. Biol.*, **15**, 91–103. PMID: 18199023.

Springer,M.S. and Gatesy,J. (2018) On the importance of homology in the age of phylogenomics. *Syst. Biodiversity*, **16**, 210–228.

Tan,G. *et al.* (2015) Current methods for automated filtering of multiple sequence alignments frequently worsen single-gene phylogenetic inference. *Syst. Biol.*, **64**, 778–791.

Warnow,T. *et al.* (2001). Absolute convergence: True trees from short sequences. In: *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01. Society for Industrial and Applied Mathematics, USA, pp. 186–195.

Xi,Z. *et al.* (2016) The impact of missing data on species tree estimation. *Mol. Biol. Evol.*, **33**, 838–860.

Zhang,C. *et al.* (2018) ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, **19**, 153.

Zhang,C. *et al.* (2020) TAPER: pinpointing errors in multiple sequence alignments despite varying rates of evolution, 12, 2145–58. https://doi.org/10.1111/2041-210X.13696. .