Information and Inference: A Journal of the IMA (2023) **12**, 1–51
https://doi.org/10.1093/imaiai/iaad032

Spectral top-down recovery of latent tree models

Yariv Aizenbud* and Ariel Jaffe
Program in Applied Mathematics, Yale University, New Haven, CT 06511, USA
*Corresponding author. Email: yariv.aizenbud@yale.edu

Meng Wang
Department of Pathology, Yale University, New Haven, CT 06511, USA

Amber Hu and Noah Amsel
Program in Applied Mathematics, Yale University, New Haven, CT 06511, USA

Boaz Nadler
Department of Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel

Joseph T. Chang
Department of Statistics, Yale University, New Haven, CT 06520, USA

and

Yuval Kluger
Program in Applied Mathematics, Yale University, New Haven, CT 06511, USA
Department of Pathology, Yale University, New Haven, CT 06511, USA
Interdepartmental Program in Computational Biology and Bioinformatics, Yale University, New Haven, CT 06511, USA

[Received on 10 December 2021; revised on 24 March 2023; accepted on 24 June 2023]

Modeling the distribution of high-dimensional data by a latent tree graphical model is a prevalent approach in multiple scientific domains. A common task is to infer the underlying tree structure, given only observations of its terminal nodes. Many algorithms for tree recovery are computationally intensive, which limits their applicability to trees of moderate size. For large trees, a common approach, termed *divide-and-conquer*, is to recover the tree structure in two steps. First, separately recover the structure of multiple, possibly random subsets of the terminal nodes. Second, merge the resulting subtrees to form a full tree. Here, we develop spectral top-down recovery (STDR), a deterministic divide-and-conquer approach to infer large latent tree models. Unlike previous methods, STDR partitions the terminal nodes in a non random way, based on the Fiedler vector of a suitable Laplacian matrix related to the observed nodes. We prove that under certain conditions, this partitioning is consistent with the tree structure. This, in turn, leads to a significantly simpler merging procedure of the small subtrees. We prove that STDR is statistically consistent and bound the number of samples required to accurately recover the tree with high probability. Using simulated data from several common tree models in phylogenetics, we demonstrate that STDR has a significant advantage in terms of runtime, with improved or similar accuracy.

Keywords: latent tree models; divide-and-conquer; spectral graph theory.

2010 Math Subject Classification: 62H22; 62M15; 05C50; 15A18.

† YA and AJ contributed equally to this work.

1. Introduction

Learning the structure of latent tree graphical models is a common task in machine learning [4, 10, 23, 42, 63] and computational biology [29, 30]. A canonical application is phylogenetics, where the task is to infer the evolutionary tree that describes the relationship between a group of biological species based on their nucleotide or protein sequences [18, 43, 48]. Depending on the application, the number of observed nodes ranges from a dozen and up to tens of thousands.

In latent tree graphical models, every node is associated with a random variable. A key assumption is that the given data corresponds to the terminal nodes of a tree, while the set of unobserved internal nodes determines its distribution. In phylogenetics, the terminal nodes are existing organisms, while the non-terminal nodes correspond to their extinct ancestors. Given a set of nucleotide or amino acid sequences as in Figure 1, the task is to recover the structure of the tree, which describes how the observed organisms evolved from their ancestors.

Many algorithms have been developed for recovering latent trees. Distance-based methods, including the classic neighbour joining (NJ) [46] and UPGMA [50], recover the tree based on a distance measure between all pairs of terminal nodes. These methods are computationally efficient and thus applicable to large trees [57], and also have theoretical recovery guarantees [5, 36]. Since the distance measure does not encapsulate all the information available from the sequences, distance-based methods may perform poorly when the amount of data is limited [61].

A different approach for tree recovery is based on spectral properties of the input data [3, 16]. Several methods work top-down, repeatedly applying spectral partitioning to the terminal nodes until each partition contains a single node [35, 64]. However, there is no theoretical guarantee that the partitions match the structure of the tree. Of direct relevance to this manuscript is the recently proposed spectral neighbour joining (SNJ) [26], which consistently recovers the tree based on a spectral criterion. Similarly to NJ, SNJ is a bottom-up method, which iteratively merges subsets of nodes to recover the tree.

Perhaps one of the most accurate approaches for tree recovery is to search for the topology that maximizes the likelihood of the observed data [18]. Since computing the likelihood for every possible topology is intractable, many methods apply a local search to iteratively increase the likelihood function [21, 44, 51, 65]. Though there is no guarantee that such a process will converge to the global maximum of the likelihood function, in many settings the resulting tree is more accurate than the one obtained by distance-based methods. The main disadvantage of likelihood-based algorithms is their slow runtime, which limits their applicability to trees of moderate size.

With the dramatic increase in the sizes of measured datasets, there is a pressing need to develop fast tree recovery algorithms, able to handle trees with tens of thousands of nodes [47, 57]. For example, the recently developed GESTALT method combines scRNA-seq readouts with CRISPR/Cas9 induced

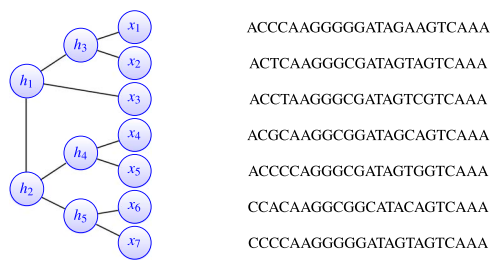


FIG. 1. A tree with $m = 7$ observed nodes. The data consist of a sequence of characters at every terminal node.

mutations to perform lineage tracing on tens of thousands of cells [45, 49]. For the multispecies coalescent model, recent works recover multiple gene trees, where each tree is composed of thousands of genes [37]. Recently, many works recovered the evolutionary history of the SARS-COV-2 virus, with over ten thousand variants [40].

Tree recovery problems with thousands of terminal nodes pose a significant computational challenge, as even distance-based methods may prove to be too slow. To improve the scalability of slow but accurate methods such as maximum likelihood, a common framework known as *divide-and-conquer* is to recover the tree by a two-step process [39, 59]: (i) infer the tree structure independently for a large number of small possibly random subsets of terminal nodes; (ii) compute the full tree by merging the small trees obtained in step (i). In *supertree* methods, the small subsets of terminal nodes in step (i) overlap. Their merging step requires optimizing a non-convex objective, which is computationally hard [25, 28]. Thus, most supertree methods circumvent global optimization problems by iterative approaches for step (ii) [55, 59]. Recently, several methods were derived to merge subtrees with disjoint terminal nodes [38, 39]. To apply these algorithms in a divide-and-conquer pipeline, the terminal nodes are partitioned according to an initial tree estimate computed by NJ. Despite these works, the problem of reconstructing large trees from limited amount of data is not yet fully resolved. In particular, there is still a need for fast and scalable approaches that also have strong recovery guarantees.

Contributions and outline In this work, we develop spectral top-down recovery (STDR), a scalable divide-and-conquer approach backed by theoretical guarantees to recover large trees. In contrast to previous methods, the partitioning of the terminal nodes in step (i) is *deterministic*. Importantly, we prove that under mild assumptions the partitions are consistent with the unobserved tree structure. The importance of this consistency is that it simplifies considerably the merging process in step (ii) of the algorithm. Since STDR is recursive, we replace the standard divide-and-conquer two step outline, with the following recursive description.

(i) Partitioning: split the terminal nodes into two subsets.

(ii) Recursive reconstruction: infer the latent tree of each subset. When the partition size falls below a given threshold τ, the tree is recovered by a user-specified algorithm. Above this threshold, the reconstruction is done by recursively applying STDR to each subset.

(iii) Merging: reconstruct the full tree by merging the two small trees.

Each of the above three steps is explained in detail in Section 3. In step (i), we apply spectral partitioning to a weighted complete graph, with nodes that correspond to the terminal nodes of the tree and weights based on a similarity measure described in Section 3.1. In Section 4.1, we prove that given an accurate estimate of these similarities, step (i) is consistent in the sense that the resulting subsets belong to two disjoint subtrees. For this proof, we derive a novel relation between latent tree models and a classic result from spectral graph theory known as Fiedler's theorem of nodal domains [19]. This theorem is important in various learning tasks such as clustering data [58], graph partitioning [12], and low dimensional embeddings [27]. To the best of our knowledge, this is the first guarantee for spectral partitioning in the setting of latent tree models.

The output of step (ii) is the inner structure of two disjoint subtrees. The task in step (iii) is to merge them into the full tree. In Section 3.4, we show that this task is equivalent to finding the root of an unrooted tree, given a reference set of one or more sequences, also known as an *outgroup*. We derive a novel spectral-based method to find the root and prove its statistical consistency in Section 4.2. This approach is of independent interest, as finding the root of a tree is a common challenge in phylogenetics [7, 8, 32]. In Section 5, we derive finite sample guarantees for a family of trees generated according to the

molecular clock model. The analysis provide useful insights into the impact of various tree parameters on the accuracy of our approach.

In Section 7, we compare the accuracy and runtime of various methods when applied to recover the full tree directly versus when used as subroutines in step (ii) of STDR. For example, Figure 6 shows the results of recovering simulated trees with 2000 terminal nodes generated according to the birth-death model [52]. As one baseline, we applied RAxML [51], one of the most popular maximum likelihood software packages in phylogenetics. With 8000 samples, RAxML took over $2\frac{1}{2}$ hours to complete. In contrast, STDR with RAxML as subroutine and a threshold $\tau = 256$ took approximately 30 minutes, about five times faster. Importantly, in this setting, the trees recovered via STDR have similar accuracy to those obtained by applying RAxML directly. These and other simulation results illustrate the potential benefit of STDR in recovering large trees.

2. Problem setup

Let $\mathscr{T}$ be an unrooted binary tree with m terminal nodes. We assume that each node of the tree has an associated discrete random variable over the alphabet $\{1,\dots,\ell\}$. We denote by $\boldsymbol{x} = (x_1,\dots,x_m)$ the vector of the random variables at the m observed terminal nodes of the tree, and by $\boldsymbol{h} = (h_1,\dots,h_{m-2})$ the random variables at the non-terminal nodes. We assume that these random variables form a Markov random field on $\mathscr{T}$. This means that given the values of its neighbours, the random variable at a node is statistically independent of the rest of the tree [9]. An edge $e(h_i,h_j)$ connecting a pair of adjacent nodes (h_i,h_j) is equipped with two transition matrices of size $\ell\times\ell$,

$$P(h_i|h_j)_{ba} = \Pr[h_i = b|h_j = a], \qquad P(h_j|h_i)_{ba} = \Pr[h_j = b|h_i = a]. \tag{2.1}$$

Note that every pair of adjacent nodes may in general have different transition matrices.

Our observed data is a matrix $X = [\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(n)}] \in \{1,\dots,\ell\}^{m\times n}$, where $\boldsymbol{x}^{(j)}$ are random i.i.d. realizations of $\boldsymbol{x} = (x_1,\dots,x_m)$. Each row in the matrix is a sequence of length n that corresponds to a terminal node in the tree, see illustration in Figure 1. For example, in phylogenetics, each row in the matrix corresponds to a different species, while each column corresponds to a different location in a DNA sequence, see [14] and references therein. Figure 1 shows an example with $m = 7$ terminal nodes and $n = 23$ observations. The support of each node is the DNA alphabet A, C, G, T, so $\ell = 4$.

Given the matrix X, the task at hand is to recover the structure of the hidden tree $\mathscr{T}$. We assume that for every pair of adjacent nodes (h_i,h_j), the corresponding $\ell\times\ell$ stochastic matrices $P(h_i|h_j)$ and $P(h_j|h_i)$ defined in (2.1) are full rank, with determinants that satisfy

$$0 < \delta < \det(P(h_i|h_j)), \det(P(h_j|h_i)) < \xi < 1. \tag{2.2}$$

Eq. (2.2) implies that the transition matrices are invertible and are not permutation matrices. This assumption is necessary for the tree's topology to be identifiable, see Proposition 3.1 in [9] and [41]. Next, to describe our approach we present several definitions related to unrooted trees, following the terminology of [60].

DEFINITION 2.1. [clan] A clan is a subset of nodes in $\mathscr{T}$ that is connected to the rest of the tree by a single edge.

DEFINITION 2.2. the root of a clanA non-terminal node h is termed the root of a clan C if $h \in C$ and it is connected to the edge that separates C from the rest of the tree.

For example, in Figure 1 h_4 and h_5 are the root nodes of the clans $C_1 = \{x_6, x_7, h_5\}$ and $C_2 = \{x_4, x_5, x_6, x_7, h_2, h_4, h_5\}$, respectively. In our work, we will sometimes refer to the clans by their terminal nodes only (e.g. $\{x_6, x_7\}$ and $\{x_4, x_5, x_6, x_7\}$ for C_1 and C_2).

DEFINITION 2.3. [adjacent clans] Let C_1 and C_2 be two disjoint subsets of terminal nodes that form two clans. If the union $C_1 \cup C_2$ forms a clan, then C_1 and C_2 are adjacent clans.

Two disjoint clans whose respective root nodes share a common neighbouring node are adjacent clans. For example, in Figure 1 the clans $C_1 = \{x_4, x_5\}$ and $C_2 = \{x_6, x_7\}$ are adjacent. Their respective root nodes h_4 and h_5 are adjacent to h_2. This observation is important for the merging step of STDR. Throughout the paper, we denote by $|C|$ the number of elements in a subset C. We denote by $\|A\|$ and $\|A\|_F$ the spectral and Frobenius norm of a matrix A.

3. A spectral top-down approach for tree reconstruction

Here we present the three steps of the STDR algorithm, as outlined in the introduction. Pseudocode for the method appears in Algorithm 1. We begin with the definition and properties of the similarity matrix and similarity graph.

Algorithm 1 STDR: Spectral Top-Down Recovery

Input: $X \in \{1, \ldots, \ell\}^{m \times n}$ A matrix containing sequences from m terminal nodes
$\tau \in \mathbb{N}$ Partition size threshold
Alg An algorithm for recovering small tree structures
Output: $\mathcal{T}$ Estimated tree

if number of terminal nodes $m \leqslant \tau$ **then**
  return Alg(X) ▷ Recover small tree structures by a user defined algorithm
end if
Compute the similarity matrix S from X via Eq. (3.2)
Compute the Fiedler vector v of S
▷ Partitioning step
Partition the terminal nodes into two subsets C_1 and C_2 by thresholding v via Eq. (3.3)
▷ Recursive reconstruction step
$\mathcal{T}_1 = \text{STDR}(X(C_1, :), \tau, \text{Alg})$
$\mathcal{T}_2 = \text{STDR}(X(C_2, :), \tau, \text{Alg})$
▷ Merging step
Compute u, the first left singular vector of $S(C_1, C_2)$
for all edges e in $\mathcal{T}_1$ **do**
  Compute the edge score $d(e)$ from u via Eq. (3.6)
end for
Insert a root node for $\mathcal{T}_1$ into the edge $e_1 = \text{argmin}_{e \in \mathcal{T}_1} d(e)$
Compute v, the first right singular vector of $S(C_1, C_2)$
for all edges e in $\mathcal{T}_2$ **do**
  Compute the edge score $d(e)$ from v via Eq. (3.6)
end for
Insert a root node for $\mathcal{T}_2$ into the edge $e_2 = \text{argmin}_{e \in \mathcal{T}_2} d(e)$
Connect the roots of $\mathcal{T}_1$ and $\mathcal{T}_2$ to construct the merged tree $\mathcal{T}$
return $\mathcal{T}$

3.1 *The pairwise similarity matrix and similarity graph*

Similar to Eq. (2.1), we define the $\ell \times \ell$ transition matrix for every pair h_i, h_j of (not necessarily adjacent) nodes by

$$P(h_i|h_j)_{ba} = \Pr[h_i = b|h_j = a].$$

Note that due to the Markov assumption, the transition matrix is multiplicative along the edges of the tree. For example in Figure 1, $P(x_1|x_2) = P(x_1|h_3)P(h_3|x_2)$. In [26], a similarity function between a pair of nodes h_i and h_j was defined as follows:

$$S(h_i, h_j) = \sqrt{\det(P(h_i|h_j))\det(P(h_j|h_i))}. \tag{3.1}$$

Similar to the transition matrix, the similarity is multiplicative along the edges of the tree and is bounded by $\delta \leq S(h_i, h_j) \leq \xi$. Thus, it exhibits an exponential decay along the tree. For any two ordered sets of terminal or non-terminal nodes $A = \{a_1, \dots a_r\}$ and $B = \{b_1, \dots b_s\}$, we denote by $S(A, B)$ a matrix of size $r \times s$, where

$$S(A, B)_{ij} = S(a_i, b_j) \quad \text{for all } 1 \leq i \leq r \text{ and} 1 \leq j \leq s.$$

To simplify notation, for the case where A and B are both equal to the full set of terminal nodes, we denote the similarity matrix by S:

$$S = S(\boldsymbol{x}, \boldsymbol{x}) \quad \text{where } \boldsymbol{x} = \{x_1, \dots, x_m\}. \tag{3.2}$$

where by definition, $S_{ii} = 1\ \forall(i)$. The matrix S is the adjacency matrix of the following graph.

DEFINITION 3.1. [Similarity graph] The similarity graph G is a complete graph whose vertices are the terminal nodes of $\mathscr{T}$. The weight assigned to every edge $e(x_i, x_j)$ is the similarity $S(x_i, x_j)$.

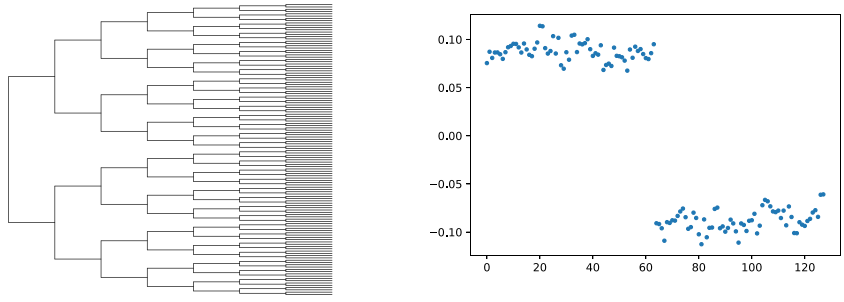
The relation between the spectral properties of G and the topology of $\mathscr{T}$ forms the theoretical basis of our approach. The following result from [26, Lemma 3.1] shows how the spectral structure of the similarity matrix S relates to the structure of the underlying tree.

LEMMA 3.1. Let A and B be a partition of the terminal nodes of an unrooted binary tree $\mathscr{T}$. The matrix $S(A, B)$ is rank-one if and only if A and B are clans of $\mathscr{T}$.

Lemma 3.1 implies that given the exact similarity matrix S, one can determine if a subset A of terminal nodes is a clan in $\mathscr{T}$ by computing the rank of $S(A, A^c)$, where $A^c = \boldsymbol{x} \setminus A$. In practice, the exact similarity matrix S is unknown. Yet, as shown in [26], a sufficiently accurate estimate $\hat{S}$, which in general is full rank, still allows to determine if a subset is a clan.

3.2 *Tree partitioning via spectral clustering*

The aim of step (i) of STDR is to partition the terminal nodes into two clans of $\mathscr{T}$. Our approach is based on the similarity graph G of Definition 3.1. One possible way to partition the graph is by the min-cut criteria. Given the exact similarity, this approach is guaranteed to yield two clans, see Lemma B.1 in



(a) Illustration of a symmetric binary tree. (b) Fielder vector of binary symmetric tree

FIG. 2. Symmetric binary tree with 128 terminal nodes. The data consists of sequences of length $n = 1000$ over the $\ell = 4$ characters of the DNA alphabet, generated according to the HKY model.

the appendix. Though the min-cut problem can be solved efficiently [58], it often leads to unbalanced partitions of the graph, with the smaller one containing 1 or 2 terminal nodes. Since one goal is to reduce the runtime of the reconstruction algorithm in step (ii), we would like to avoid imbalanced partitions. To this end, we propose to partition the terminal nodes via a spectral approach based on the Fiedler vector.

DEFINITION 3.2. Graph Laplacian and Fiedler vector. The Laplacian matrix of a graph G with a symmetric weight matrix W is given by $L_G = D - W$, where D is a diagonal matrix with $D_{ii} = \sum_j W(x_i, x_j)$. The Fiedler vector is the eigenvector of L_G that corresponds to the second smallest eigenvalue.

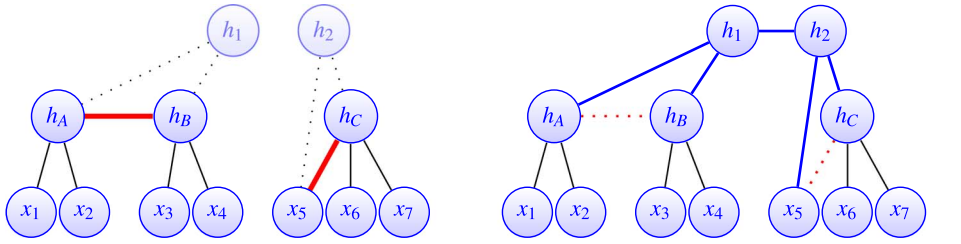
In the STDR algorithm, we use the Fiedler vector v of the similarity graph G to partition the terminal nodes into two subsets C_1 and C_2 (Algorithm 1, line 6), as follows:

$$C_1 = \{i; v(i) \geq 0\}, \qquad C_2 = \{i; v(i) < 0\}. \tag{3.3}$$

Importantly, in Section 4.1 we prove that partitioning the nodes of G via Eq. (3.3) yields two clans of the underlying tree $\mathscr{T}$. To illustrate this point, we created a tree graphical model from a symmetric binary tree with $m = 128$ nodes, see Figure 2(a). The transition matrices between adjacent nodes are all identical and were chosen according to the HKY model [24]. We used this model to generate a dataset of nucleotide sequences of length $n = 1000$. Figure 2(b) shows the Fiedler vector of the similarity graph estimated from the dataset. Here, the Fiedler vector exhibits a single dominant gap, and partitioning the terminal nodes by Eq. (3.3) yields two sets C_1 and C_2 which are indeed clans of $\mathscr{T}$. A similar example is shown in the appendix for a tree generated according to the coalescent model. In Section 5.1, we derive a lower bound for the number of samples sufficient to obtain two clans with high probability.

3.3 *Recursive reconstruction step*

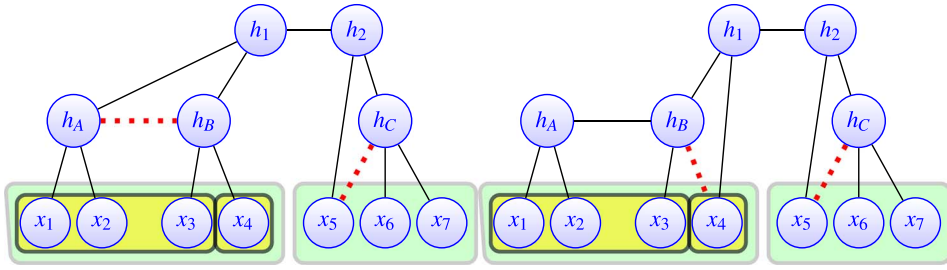
Step (i) of STDR outputs two sets of terminal nodes C_1 and C_2. Under certain conditions defined in Section 4.1, these are guaranteed to be two clans in the tree $\mathscr{T}$. The next task is to construct trees $\mathscr{T}_1$ and $\mathscr{T}_2$ that describe their latent internal structure. If $|C_1| > \tau$, then $\mathscr{T}_1$ is recovered by recursively applying the three steps of STDR to C_1. When $|C_1| \leq \tau$, the input is small enough that we consider it tractable to use a direct method for tree reconstruction, even a slow one like maximum likelihood.



(a) Two unrooted trees. The placeholder edges are marked in red.

(b) The merging process is completed by connecting the two root nodes h_1, h_2.

FIG. 3. Merging example.



(a) Placeholder edges set to $e(h_A, h_B)$ and $e(h_C, x_5)$.

(b) Placeholder edges set to $e(h_B, x_4)$ and $e(h_C, x_5)$.

FIG. 4. Different choices of placeholder edges result in a different merged trees.

3.4 *Merging disjoint subtrees*

The output of step (ii) of STDR consists of the internal *unrooted* tree structures $\mathscr{T}_1$ and $\mathscr{T}_2$ of two subsets of terminal nodes C_1 and C_2. Assuming steps (i) and (ii) were successful, then C_1 and C_2 are adjacent clans, and $\mathscr{T}_1$ and $\mathscr{T}_2$ are indeed their correct internal structures. The remaining challenge in step (iii) is to recover the full tree $\mathscr{T}$ by correctly merging $\mathscr{T}_1$ and $\mathscr{T}_2$.

Since $\mathscr{T}_1$ and $\mathscr{T}_2$ are unrooted binary trees, to merge them it is necessary to add a root node to each of them. Adding a connecting edge between the two root nodes yields a binary unrooted tree and completes the merging process, see Figure 3 for an illustration. Adding a root node to a given subtree requires finding which one of its edges is the the "placeholder edge" (illustrated in red in Figure 3(a)). Subsequently, the placeholder edge is replaced with two edges connected to the root node. Importantly, as shown in Figure 3, different choices for the placeholder edge in either $\mathscr{T}_1$ or $\mathscr{T}_2$ yield a merged tree with a different topology.

Thus, merging $\mathscr{T}_1$ and $\mathscr{T}_2$ reduces to the task of identifying the correct 'placeholder edge'. Here, we derive a novel spectral method for finding these edges. To the best of our knowledge, our approach for merging subtrees is new and may be of independent interest for other applications, such as rooting unrooted trees [7, 8, 32]. In the following lemma, whose proof is in Appendix C, we describe a property of the placeholder edge that motivates our approach.

LEMMA 3.2. Let C_1 be a set of terminal nodes that forms a clan in $\mathscr{T}$, and let $\mathscr{T}_1$ be the internal structure of C_1. An edge $e \in \mathscr{T}_1$ is the correct placeholder edge if and only if it partitions C_1 into two sets $A(e), B(e)$, such that both form clans in $\mathscr{T}$.

Lemma 3.2 is illustrated in Figure 3(a). The edge $e(h_A, h_B)$ divides the left subtree into the clans $\{x_1, x_2\}$ and $\{x_3, x_4\}$. These subsets also form clans in the full tree depicted in Figure 3(b).

Next, using Lemma 3.2, we derive a spectral characterization of the correct placeholder edge. Recall that by Lemma 3.1, the matrix $S(C_1, C_2) \in \mathbb{R}^{|C_1|\times|C_2|}$, is rank one. Thus,

$$S(C_1, C_2) = u\sigma v^T, \quad \text{where} \quad \|v\| = \|u\| = 1, \quad \text{and} \quad \sigma > 0. \tag{3.4}$$

Given a placeholder edge e and its corresponding partition of terminal nodes $A(e)$ and $B(e)$, we denote by $u_{A(e)}, u_{B(e)}$ the entries of u that correspond to $A(e)$ and $B(e)$, respectively. The following lemma, proven in Appendix C, characterizes the correct placeholder edge in terms of $u_{A(e)}$ and $u_{B(e)}$.

LEMMA 3.3. An edge e is the correct placeholder edge of $\mathscr{T}_1$ if and only if there exists a constant α such that

$$S(A(e), B(e)) = u_{A(e)} \alpha u_{B(e)}^T. \tag{3.5}$$

In practice, we can only compute estimates $S, \hat{u}$ of the similarity S and vector u. Motivated by Lemma 3.3, we propose to determine the placeholder edge e^* by minimizing the following score function,

$$e^* = \operatorname*{argmin}_e d(e) = \operatorname*{argmin}_e \frac{1}{\|\hat{S}(A(e), B(e))\|_F} \min_\alpha \|\hat{S}(A(e), B(e)) - \hat{u}_{A(e)} \alpha \hat{u}_{B(e)}^T\|_F. \tag{3.6}$$

The normalizing factor $\|S(A(e), B(e))\|_F$ is added since the size of $S(A(e), B(e))$ changes for every edge e. Note that given the exact matrix S, at the correct placeholder edge $d(e^*) = 0$. In Section 5.2, we derive a sufficient number of samples that guarantee recovery of the correct placeholder edge by Eq. (3.6) with high probability.

4. Correct tree recovery of STDR

In this section, we consider the population setting where the similarity matrix S is known. Under the assumption that the subroutine Alg (See Alg. 1) accurately recovers the subtrees of size τ, we prove that STDR correctly recovers the full tree. We do so by analyzing the partitioning step (i) and the merging step (iii) of STDR. Our key results are Theorem 4.2, which states that step (i) is guaranteed to yield disjoint clans, and Theorem 4.5, which states that given accurate trees for two clans, step (iii) recovers the exact structure of the full tree. Combining these two results directly yields the following theorem establishing the correctness of STDR in the population setting.

THEOREM 4.1. Given an exact similarity matrix S, and assuming that the subroutine Alg correctly recovers the internal structure of its input, STDR recovers the exact latent tree $\mathscr{T}$.

4.1 *Consistency of the partition step*

The following theorem proves that given the exact similarity matrix, partitioning the terminal nodes of the tree by thresholding the Fiedler vector as described in Section 3.1 yields two adjacent clans.

THEOREM 4.2. Let G be the similarity graph of a binary tree $\mathscr{T}$. Denote by v the Fiedler vector of G and by $\{C_1, C_2\}$ a partition of the terminal nodes according to the sign pattern of v as in Eq. (3.3). Then C_1, C_2 are adjacent clans in $\mathscr{T}$.

Before proving Theorem 4.2, we would like to put its novelty into the context of related results. A result similar in nature to Theorem 4.2 was proved for hierarchical block models (HBM) [6] where the underlying block structure of a given connectivity matrix is recovered by recursive partitioning according to its Fiedler vector. Their statistical guarantee, however, requires additional assumptions on the parameters of the hierarchical block model. Theorem 4.2, in contrast, holds for any tree structure and parameters. A different distance-based approach for tree partitioning was derived in chapter 4 of [20]. This approach is guaranteed to yield two clans given the exact distance matrix between terminal nodes. In Appendix D, we show empirically that our similarity based approach is more robust than the distance based approach, specifically in cases where the number of samples is limited.

For the proof of Theorem 4.2, we present several preliminaries on graphs. First, we define the Schur complement of a matrix, which plays an important role in graph theory [13].

DEFINITION 4.3. Schur complement. Let A, B, C and D be matrices of dimensions $p \times p, p \times q, q \times p$ and $q \times q$, respectively. Assume D is invertible and consider the matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix},$$

of size $(p+q) \times (p+q)$. The Schur complement of D with respect to M is the $p \times p$ matrix

$$M/D = A - BD^{-1}C.$$

Let H be a graph with a set of nodes V and Laplacian matrix L. We denote by L_R the principal sub-matrix of L that corresponds to a subset of nodes $R \subset V$. The Schur complement of L_R with respect to L yields the Laplacian of a different graph, with $|V \setminus R|$ nodes [11, 13]. We denote this Laplacian matrix by $L_{H/R}$. The rows and columns of $L_{H/R}$ correspond to vertices of H that are not in R. When the graph is a tree $\mathscr{T}$, and R is the set of its non-terminal nodes, then $L_{\mathscr{T}/R}$ is the Laplacian of a complete graph G whose nodes are the terminal nodes of $\mathscr{T}$.

Equipped with these definitions, we proceed to the proof of Theorem 4.2. The proof consists of two parts, that correspond to Theorem 4.4 and Lemma 4.1. Theorem 4.4, which is a rephrase of Theorem 3.3 of [54], shows that one can partition the terminal nodes of a tree $\mathscr{T}$ into two clans via the Fiedler vector of $L_{\mathscr{T}/R}$, where R is the set of the tree's internal nodes.

THEOREM 4.4. [54],Theorem 3.3[1]. Let $\mathscr{T}$ be a tree with a node set V and a subset of non terminal nodes $R \subset V$. We denote by $L_{\mathscr{T}}$ the Laplacian of $\mathscr{T}$, by L_R the sub-matrix of $L_{\mathscr{T}}$ that corresponds to the nodes in R, and by $L_{\mathscr{T}/R}$ the Laplacian of a graph G obtained by Schur complement of L_R with respect $L_{\mathscr{T}}$. Let v be the Fiedler vector of G, and C_1, C_2 the following partition of the terminal nodes,

$$C_1 = \{i \in V \setminus R;\ v(i) \leq 0\}, \qquad C_2 = \{j \in V \setminus R;\ v(j) > 0\}.$$

Then C_1 and C_2 are adjacent clans in $\mathscr{T}$.

Theorem 4.4, however, is not directly applicable to our setting, since computing $L_{\mathscr{T}/R}$ requires knowledge of the unknown similarities between all nodes of $\mathscr{T}$, including its unobserved nodes. Here, we derive Lemma 4.1 that shows that for any tree $\mathscr{T}$, there is a *twin tree* $\tilde{\mathscr{T}}$ with the same topology,

[1] For clarity, we rephrased the theorem from [54] according to our terminology.

such that $L_{\widetilde{\mathcal{T}}/R} = L_G$. This result, proven in appendix E, provides the critical missing link required for inference of the latent tree from the similarity matrix, which can be estimated from observed data.

LEMMA 4.1. Let $\mathcal{T}$ be a tree with a set of internal nodes R. Let G be the similarity graph of $\mathcal{T}$. Then there is a tree $\widetilde{\mathcal{T}}$ with the same topology as $\mathcal{T}$ but different edge weights, such that

$$L_G = L_{\widetilde{\mathcal{T}}/R}.$$

Combining Lemma 4.1 with Theorem 4.4 yields the following proof of Theorem 4.2.

Proof of Theorem 4.2. Let L_G be the Laplacian matrix of the similarity graph G. By Lemma 4.1 there is a tree $\widetilde{\mathcal{T}}$ with the same topology as $\mathcal{T}$ such that $L_G = L_{\widetilde{T}/R}$. By Theorem 4.4, partitioning the terminal nodes of $\widetilde{\mathcal{T}}$ according to the sign pattern of the Fiedler vector of $L_{\widetilde{\mathcal{T}}/R}$ yields adjacent clans in $\widetilde{\mathcal{T}}$. Since $L_G = L_{\widetilde{\mathcal{T}}/R}$ and $\widetilde{\mathcal{T}}$ has the same topology as $\mathcal{T}$, it follows that partitioning the terminal nodes of $\mathcal{T}$ according to the Fiedler vector of L_G yields adjacent clans in $\mathcal{T}$. □

4.2 *Correctness of the merging step*

Step (iii) of STDR merges the two subtrees, $\mathcal{T}_1$ and $\mathcal{T}_2$, that were constructed from the two disjoint subsets of terminal nodes C_1 and C_2. As described in Section 3, this step is done by finding for each tree its placeholder edge as the edge with the smallest score $d(e)$, Eq. (3.6). Here, we prove that this merging step is correct, under the following two assumptions on its input (the output of steps (i) and (ii)): the two subtrees $\mathcal{T}_1, \mathcal{T}_2$ correspond to adjacent clans in $\mathcal{T}$ and their internal structure was recovered correctly.

THEOREM 4.5. Let C_1 and C_2 be the terminal nodes of two adjacent clans that partition a tree $\mathcal{T}$. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the internal structures of these clans. Then given the exact similarity matrix $S(C_1, C_2)$, minimizing the criterion in Eq. (3.6) yields the correct placeholder edge.

Proof. By Lemma 3.3, for the correct placeholder edge e^* there exists an $\alpha \in \mathbb{R}$ such that

$$S(A(e^*), B(e^*)) = u_{A(e^*)} \alpha u_{B(e^*)}^T.$$

Hence $d(e^*) = 0$. If e is an incorrect placeholder edge, then again according to Lemma 3.3 there is no constant α that satisfies the equation, and hence $d(e) > 0$ which implies $e^* = \operatorname{argmin} d(e)$. □

5. Finite sample guarantees for STDR

In practice, the similarity matrix S is unknown, and an estimate $\hat{S}$ is computed from sequences of length n. In this section we show that STDR is still able to correctly recover the tree provided that $\hat{S}$ is sufficiently close to S. Specifically, in Sections 5.1 and 5.2, we derive an explicit formula for the number of samples n that suffice to guarantee, with high probability, the success of a single step of partitioning and merging. In Section 5.3, we extend the results to multiple iterations and compare them to guarantees derived for other tree recovery algorithms.

For simplicity, in the finite sample analysis, we assume the Jukes-Cantor (JC) model of sequence evolution, where each transition matrix is parameterized by a single mutation rate $\theta(i,j)$:

$$P(h_i|h_j)_{ba} = P[h_i = b|h_j = a] = \begin{cases} 1 - \theta(i,j) & a = b \\ \theta(i,j)/(\ell - 1) & a \neq b. \end{cases} \tag{5.1}$$

According to this model, the similarity between adjacent nodes defined in Eq. (3.1) simplifies to

$$S(h_i, h_j) = \left(1 - \frac{\ell}{\ell - 1}\theta(i,j)\right)^{\ell-1}.$$

By Eq. (2.2) the similarity is strictly positive and hence $\theta(i,j) < (\ell-1)/\ell$. We remark that our analysis can be extended, under minor additional assumptions to more general models of evolution as in [26, Lemma 4.8).

5.1 *Finite sample guarantees for the partitioning step*

We derive an explicit expression for the number of samples n that guarantee with high probability, that the partitioning step would yield two clans. In our derivation, we assume that the similarity matrix S satisfies the hierarchical constant block model (CBM) addressed in [6]. We assume there is a hierarchy of subsets A_i and B_i (for a formal definition of hierarchical subsets/clusters see Definition 1 in [6]), and that for each subset there is a (different) constant c such that

$$\begin{aligned} &S(x,y) = c \qquad \forall (x,y) \in A_i \times B_i, \\ &S(x,y) > c \qquad \forall (x,y) \in A_i \times A_i \quad \text{and} \quad \forall (x,y) \in B_i \times B_i. \end{aligned} \tag{5.2}$$

In phylogenetics, this assumption is satisfied in the molecular clock model [34], and we will thus use the two terms interchangeably. In the molecular block model, the probability of mutation between adjacent nodes is determined by two factors: (i) the edge length between them and (ii) a mutation rate matrix that is constant throughout the tree. The structure of the rate matrix is determined by the choice of evolutionary model, such as Jukes-Cantor. In addition, the path length between all terminal nodes and the root is constant. This implies that for every ancestor h (internal node) the similarity between the terminal nodes A_i on the left of h and the nodes B_i on the right of h is constant as in Eq. (5.2). For the hierarchy of partitions, we denote by η the maximum over all partitions C of the ratio between the size of left and right parts A_i, B_i.

$$\eta = \max_i \{|A_i|/|B_i|, |B_i|/|A_i|\}. \tag{5.3}$$

This factor serves as a measure for tree's *imbalance*. For example, for a balanced binary tree, η is equal to 1. On the other hand, for a tree similar to a caterpillar that satisfies the CBM assumptions $\eta = m - 1$, see illustration in appendix G. We denote by $r(\mathscr{T})$ the diameter of $\mathscr{T}$, which is the maximal distance between pairs of terminal nodes,

$$r(\mathscr{T}) = \max_{i,j}(-\log S(x_i, x_j)). \tag{5.4}$$

Finally, we denote by $h(\mathscr{T})$ the depth of $\mathscr{T}$ as defined in [15]:

DEFINITION 5.1. Let $\mathscr{T}_1, \mathscr{T}_2$ be two rooted subtrees with respective roots h_1, h_2 obtained by removing an edge $e(h_1, h_2)$ from $\mathscr{T}$. Let $d_1(e), d_2(e)$ be the distances $-\log S(h_1, x_i)$ and $-\log S(h_2, x_j)$ from h_1, h_2 to the *closest* leaves x_i and x_j in $\mathscr{T}_1, \mathscr{T}_2$, respectively. Then

$$h(\mathscr{T}) = \max_e \max\{d_1(e), d_2(e)\}. \tag{5.5}$$

Note that $h(\mathscr{T}) < r(\mathscr{T})$ as the maximal distance between terminal nodes is larger than any distance between a pair of terminal and non terminal nodes. The following theorem bounds the number of samples n by the properties of the tree defined in Eqs (5.3),(5.4) and (5.5).

THEOREM 5.2. Let $\mathscr{T}$ be a Jukes-Cantor evolutionary tree with m terminal nodes, and a similarity matrix S that satisfies the assumptions made for the CBM. If the number of samples n satisfies

$$n \geq 4 \ln \left(\frac{2m^2}{\epsilon} \right) \eta \ell^2 m (\sqrt{m}+1)^2 e^{2r(\mathscr{T})} \max \left\{ 1, \frac{(1+\eta)^2}{(e^{r(\mathscr{T})-h(\mathscr{T})}-1)^2} \right\},$$

then with probability at least $1-\epsilon$, STDR partitions the terminal nodes into two clans.

REMARK 5.1. For the CBM model, $h(\mathscr{T}) = r(\mathscr{T})/2$. Thus, a simpler (though a bit weaker) guarantee is that the partitioning yields two trees if,

$$n \geq 4 \ln \left(\frac{2m^2}{\epsilon} \right) \eta \ell^2 m (\sqrt{m}+1)^2 e^{2r(\mathscr{T})} \frac{(1+\eta)^2}{(1-\xi)}.$$

To prove Theorem 5.2, we derive a bound on the error in the estimate $\hat{S}$ that the partitioning step can tolerate.

LEMMA 5.1. Assume a tree with m terminal nodes generated according to the molecular clock model. If the estimate $\hat{S}$ of its similarity matrix satisfies

$$\|S-\hat{S}\| \leq \frac{\sqrt{m} e^{-r(\mathscr{T})}}{\sqrt{\eta} 2^{3/2} (\sqrt{m}+1)} \min \left\{ 1, \frac{1}{1+\eta} \left(e^{r(\mathscr{T})-h(\mathscr{T})} - 1 \right) \right\}, \tag{5.6}$$

then STDR correctly partitions the terminal nodes into two clans.

To prove Lemma 5.1, we use the following lemma regarding the spectrum of the Laplacian. This Lemma is a reformulation of lemma 7 from [6] and its proof, which address the spectrum of the CBM.

LEMMA 5.2. Let $\mathscr{T}$ be a tree with m terminal nodes generated according to the molecular clock model, and let L be the Laplacian of its similarity graph. We denote by A, B a partition of the terminal nodes such that the corresponding trees $\mathscr{T}_A, \mathscr{T}_B$ also satisfy the molecular clock model. The first, second and third smallest eigenvalues of L satisfy

$$\lambda_1 = 0, \qquad \lambda_2 = m e^{-r(\mathscr{T})}, \qquad \lambda_3 \geq \frac{m}{1+\eta} \left(\eta e^{-r(\mathscr{T})} + e^{-h(\mathscr{T})} \right).$$

Partitioning the terminal nodes according to the sign pattern of the eigenvector v_2 yields the subsets A, B. In addition, the elements of v_2 are bounded away from 0 such that $|v_2(i)| \geq \sqrt{\frac{1}{m\eta}}$ for all i.

Proof of Lemma 5.1. Let L and $\hat{L}$ be two symmetric matrices and let v_i and $\hat{v}_i$ be their i-th eigenvectors, respectively. A variant of the Davis-Kahan theorem for perturbation of eigenvectors (see Theorem 2 of

[62]) gives

$$\|v_i - \hat{v}_i\| \leq 2^{3/2} \frac{\|L - \hat{L}\|}{\gamma_i}. \tag{5.7}$$

where $\gamma_i = \min\{|\lambda_i - \lambda_{i+1}|, |\lambda_i - \lambda_{i-1}|\}$ is the eigengap. We apply the theorem to the Laplacian matrix $L = D - S$ (see Definition 3.2), and its Fiedler vector v_2. The spectral norm $\|L - \hat{L}\|$ can be bounded by,

$$\begin{aligned}\|L - \hat{L}\| &\leq \|D - \hat{D}\| + \|S - \hat{S}\| = \max_i \left| \sum_k (S_{ik} - \hat{S}_{ik}) \right| + \|S - \hat{S}\| \\ &\leq \max_i \sum_k |S_{ik} - \hat{S}_{ik}| + \|S - \hat{S}\| \leq \|S - \hat{S}\|_\infty + \|S - \hat{S}\| \leq\leq (\sqrt{m} + 1)\|S - \hat{S}\|, \end{aligned} \tag{5.8}$$

where the last inequality follows from a general bound on the infinity norm. Substituting (5.8) into (5.7) yields

$$\|v_2 - \hat{v}_2\| \leq 2^{3/2}(\sqrt{m} + 1) \frac{\|S - \hat{S}\|}{\gamma_2}. \tag{5.9}$$

From Lemma 5.2 it follows that the spectral gap γ_2 is bounded by,

$$\gamma_2 = \min(\lambda_2 - \lambda_1, \lambda_3 - \lambda_2) \geq m e^{-r(\mathscr{T})} \min\left\{1, \frac{1}{1+\eta}\left(e^{r(\mathscr{T}) - h(\mathscr{T})} - 1\right)\right\}. \tag{5.10}$$

Combining Eqs (5.9) and (5.10) proves that if

$$\|S - \hat{S}\| \leq \frac{\sqrt{m} e^{-r(\mathscr{T})}}{\sqrt{\eta} 2^{3/2}(\sqrt{m} + 1)} \min\left\{1, \frac{1}{1+\eta}\left(e^{r(\mathscr{T}) - h(\mathscr{T})} - 1\right)\right\} \tag{5.11}$$

then $\|v_2 - \hat{v}_2\| < 1/\sqrt{\eta m}$, which implies $\|v_2 - \hat{v}_2\|_\infty < 1/\sqrt{\eta m}$. Thus, by Lemma 5.2 $\operatorname{sign}(v_i) = \operatorname{sign}(\hat{v}_i)$ for each $i \in [m]$. Hence, partitioning the terminal nodes according to $\operatorname{sign}(\hat{v}_2)$ or $\operatorname{sign}(v_2)$ yields the same result. As we proved in Theorem 4.2, the resulting subsets are clans of the tree. □

Next, we prove Theorem 5.2 under the additional assumption of the Jukes-Cantor model. The theorem is proved by combining Lemma 5.1 with a concentration bound on the similarity matrix estimate $\hat{S}$, derived in [26].

Proof of Theorem 5.2. From Lemma 4.7 of [26], under the JC model of evolution,

$$P\left(\|\hat{S} - S\| \leq t\right) \geq 1 - 2m^2 \exp\left(-\frac{2nt^2}{\ell^2 m^2}\right). \tag{5.12}$$

Setting t to the right-hand side of (5.11) yields that if

$$n \geq 4 \ln\left(\frac{2m^2}{\epsilon}\right) \eta \ell^2 m (\sqrt{m} + 1)^2 e^{2r(t)} \max\left\{1, \frac{(1+\eta)^2}{(e^{r(\mathscr{T}) - h(\mathscr{T})} - 1)^2}\right\},$$

the requirements of Lemma 5.1 are satisfied with probability at least $1-\varepsilon$, which concludes the proof. □

5.2 *Merging step of STDR*

We derive finite sample bounds for the merging step of STDR. In contrast to the partitioning step, the guarantees for the merging step, presented in the following theorem, hold for any tree topology. For simplicity, Theorem 5.3 derives a guarantee for finding the correct placeholder edge in $\mathscr{T}_1$.

THEOREM 5.3. Let $\mathscr{T}$ be a tree with m terminal nodes, which consists of two subtrees $\mathscr{T}_1, \mathscr{T}_2$ with terminal nodes C_1 and C_2, respectively. Let $\{A, B\}$ be the partition of C_1 induced by the correct placeholder edge e^*, and let $\mathscr{D} = \min\{\|S(A,B)\|_F, \|S(C_1,C_2)\|_F\}$. For any $\varepsilon > 0$, if the number of samples n satisfies

$$n \geq 8\ell^2 m^3 \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1+10\sqrt{2}}{\mathscr{D}^3}\right)^2 \left(\frac{\xi^4}{\delta^8(1-\xi^2)^2}\right) \log\left(\frac{2m^2}{\varepsilon}\right), \tag{5.13}$$

then STDR finds the correct placeholder edge in $\mathscr{T}_1$ with probability at least $1-\varepsilon$. A similar guarantee can be derived for finidng the correct placeholder edge in $\mathscr{T}_2$.

Theorem 5.2 provides a theoretical guarantee for the partitioning step under the assumption of the molecular clock. For this setting, we can lower bound the value of $\mathscr{D}$ via the following lemma,

LEMMA 5.3. Under the assumption of the molecular clock model, the value of $\mathscr{D}$ is lower bounded by

$$\mathscr{D} \geq \exp(-r(\mathscr{T})) \max\left\{1, \frac{\sqrt{\eta}m}{(1+\eta)^2}\right\}.$$

Combining Lemma 5.3 with Theorem 5.3 yields that under the added assumptions of the molecular clock model, the number of observations sufficient to correctly merging two trees is of the order of

$$n = \mathscr{O}\Big(m \exp\big(2r(\mathscr{T})\big) \min\{(1+\eta)^3, m^2\}\Big).$$

which implies a linear dependency on m for relatively balanced trees, but might require $\mathscr{O}(m^3)$ for highly imbalanced trees for which $\eta = \mathscr{O}(m)$. In Appendix G, we provide an illustration and a short explanation as to the value of $\mathscr{D}$.

Our proof of Theorem 5.3 consists of three steps: (i) In Lemma 5.4 we derive a lower bound on the score $d(e)$ of an edge e that is not the correct placeholder edge; (ii) Lemma 5.7 provides a sufficient condition on the accuracy of the similarity matrix estimate $\hat{S}$ that guarantees the merging step will yield the correct placeholder edge; (iii) For the JC model, we derive an expression for the number of samples required for the condition in Lemma 5.7 to hold with high probability.

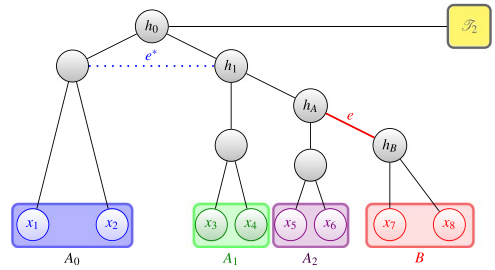


FIG. 5. Bounding the score $d(e)$ for an incorrect placeholder edge in $\mathscr{T}_1$. The correct placeholder edge $e^* \in \mathscr{T}_1$ is marked by a dotted blue line. The incorrect placeholder edge e, which partitions the terminal node to subsets $A(e)$ and $B(e)$, is marked by a thick red line. The two non-terminal nodes on the path between the correct and incorrect edges are denoted by $h_1, h_2 = h_A$ and the root node of C_1 is denoted by h_0. The subset of terminal nodes closest to h_i is denoted by A_i.

STEP 1: A LOWER BOUND ON THE SCORE $d(e)$ FOR INCORRECT EDGES. In Section 4.2, we showed that $d(e) = 0$ if and only if e is the correct placeholder edge. Here, for the exact similarity matrix S we derive a lower bound on $d(e)$, if e is an incorrect placeholder edge in $\mathscr{T}_1$.

LEMMA 5.4. Let $\mathscr{T}$ be a tree that consists of two subtrees $\mathscr{T}_1, \mathscr{T}_2$, and let $e \in \mathscr{T}_1$ be an edge that is not the correct placeholder edge. Then,

$$d(e) \geq \begin{cases} \frac{(\sqrt{2}\delta)^{\log m}\delta^3(1-\xi^2)}{2\sqrt{m}\xi^2} & \delta^2 \leq 0.5 \\ \frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2} & \delta^2 > 0.5. \end{cases}$$

For the proof of Lemma 5.4, we introduce new notation, illustrated in Figure 5. We denote by $e^* \in \mathscr{T}_1$ the correct placeholder edge, and by $e \in \mathscr{T}_1$ an arbitrary incorrect placeholder edge. The edge e splits the terminal nodes of $\mathscr{T}_1$ into A and B and has endpoints h_A and h_B. We denote by $h_0, \ldots, h_N$ the non terminal nodes on the path between the root node of $\mathscr{T}_1$, denoted h_0, and $h_A = h_N$. We partition the terminal nodes in A to $N+1$ subsets $A_0, \ldots, A_N$ according to $h_0, \ldots, h_N$ as follows: Every node in A is assigned to the closest non terminal node on the path between $h_0, \ldots, h_N$. In the proof of Lemma 5.4, we use the following auxiliary lemma, proven in appendix F.

LEMMA 5.5. Let $R_i = S(h_0, h_i)^2$. For any $1 \leq i \leq N-1$ and $1 \leq k \leq (N-i)$ we have

$$\min_{\beta} \frac{(1-\beta R_i)^2\|S(A_i,B)\|_F^2 + (1-\beta R_{i+k})^2\|S(A_{i+k},B)\|_F^2}{\|S(A_i,B)\|_F^2 + \|S(A_{i+k},B)\|_F^2} \geq \begin{cases} \frac{(2\delta^2)^{\log m}\delta^{2(k+1)}(1-\xi^2)^2}{4m\xi^4} & \delta^2 \leq 0.5 \\ \frac{\delta^{2(k+2)}(1-\xi^2)^2}{2m\xi^4} & \delta^2 > 0.5 \end{cases} \tag{5.14}$$

Proof of Lemma 5.4 The proof consists of the following steps: (i) we express the score $d(e)$ defined in Eq. (3.6) in terms of $\|S(A_0,B)\|_F, \ldots \|S(A_N,B)\|_F$. The new expression is given in Eq. (5.18). (ii) In Eq. (5.19) we derive a lower bound on $d(e)$ in terms of two consecutive terms $\|S(A_i,B)\|_F$ and $\|S(A_{i+1},B)\|_F$. (iii) In Lemma 5.5 we combine Eq. (5.19) with a bound on $\|S(A_i,B)\|_F$ and $\|S(A_{i+1},B)\|_F$ to conclude the proof.

First, we express the numerator of $d(e)$ in Eq. (3.6) in terms of $S(A_0,B), \ldots, S(A_N,B)$. Since h_0 separates the terminal nodes C_1 and C_2 of the subtrees $\mathscr{T}_1$ and $\mathscr{T}_2$, by the multiplicative property of the

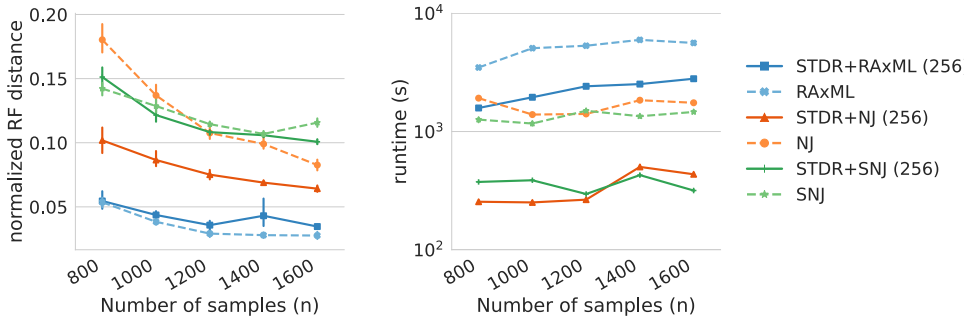


FIG. 6. A birth-death tree with $m = 2048$ terminal nodes. The mean and standard deviation of the normalized RF distance (left) between the reconstructed tree and the input tree and of the runtime (right) are shown for each method over five independent runs.

similarity we have,

$$S(C_1, C_2) = S(C_1, h_0)S(h_0, C_2) = u\sigma v^T \qquad \|u\| = \|v\| = 1.$$

Let $\bar{\beta}$ be the proportionality constant between u and $S(C_1, h_0)$ such that $u = \bar{\beta}S(C_1, h_0)$. Recall that u_A, u_B in Eq. (3.6) are the entries in u that correspond to A and B, respectively. Partitioning u into u_A and u_B and partitioning $S(C_1, h_0)$ into $S(A, h_0)$ and $S(B, h_0)$ gives

$$u_A = \bar{\beta}S(A, h_0), \qquad u_B = \bar{\beta}S(B, h_0).$$

It follows that, for any $\alpha \in \mathbb{R}$,

$$S(A, B) - u_A \alpha u_B^T = S(A, B) - \alpha\bar{\beta}^2 S(A, h_0)S(h_0, B) = S(A, B) - \beta S(A, h_0)S(h_0, B),$$

where $\beta = \bar{\beta}^2\alpha$. Next, we split the matrix $S(A, B)$ into the submatrices $S(A_0, B), S(A_1, B), \ldots, S(A_N, B)$. Similarly, we split $S(A, h_0)$ into the components $S(A_0, h_0), S(A_1, h_0), \ldots, S(A_N, h_0)$. This gives

$$S(A, B) - \beta S(A, h_0)S(h_0, B) = \begin{bmatrix} S(A_0, B) \\ S(A_1, B) \\ \vdots \\ S(A_N, B) \end{bmatrix} - \beta \begin{bmatrix} S(A_0, h_0) \\ S(A_1, h_0) \\ \vdots \\ S(A_N, h_0) \end{bmatrix} S(h_0, B). \tag{5.15}$$

We show that the matrix $S(A_i, h_0)S(h_0, B)$ on the right side of Eq. (5.15), is proportional to $S(A_i, B)$ with the proportionality constant $R_i = S(h_0, h_i)^2$.

$$\begin{aligned} R_i S(A_i, B) &= S(h_0, h_i)^2 S(A_i, h_i)S(h_i, B) \\ &= S(A_i, h_i)S(h_i, h_0)\ S(h_0, h_i)S(h_i, B) = S(A_i, h_0)S(h_0, B). \end{aligned} \tag{5.16}$$

Inserting (5.16) into (5.15) gives

$$\begin{bmatrix} S(A_0,B) \\ S(A_1,B) \\ \vdots \\ S(A_N,B) \end{bmatrix} - \beta \begin{bmatrix} R_0 S(A_0,B) \\ R_1 S(A_1,B) \\ \vdots \\ R_N S(A_N,B) \end{bmatrix} = \begin{bmatrix} (1-\beta R_0) S(A_0,B) \\ (1-\beta R_1) S(A_1,B) \\ \vdots \\ (1-\beta R_N) S(A_N,B) \end{bmatrix}.$$

Thus, the score in Eq. (3.6) is equivalent to

$$d^2(e) = \frac{1}{\|S(A,B)\|_F^2} \min_{\beta} \sum_{i=0}^{N} (1-\beta R_i)^2 \|S(A_i,B)\|_F^2. \tag{5.17}$$

Since $\|S(A,B)\|_F^2 = \sum_{i=0}^{N} \|S(A_i,B)\|_F^2$, we can rewrite Eq. (5.17) as follows,

$$d^2(e) = \min_{\beta} \frac{\sum_{i=0}^{N} (1-\beta R_i)^2 \|S(A_i,B)\|_F^2}{\sum_{i=0}^{N} \|S(A_i,B)\|_F^2}. \tag{5.18}$$

Next, the following lemma, proven in Appendix F, bounds the ratio of such two sums.

Lemma. For two series of positive numbers $a_i, b_i > 0$ we have

$$\frac{\sum a_i}{\sum b_i} \geq \min_{i \neq j; |i-j| \leq 2} \frac{a_i + a_j}{b_i + b_j}.$$

Applying the lemma to Eq. (5.18) yields

$$d^2(e) \geq \min_{0 \leq i \leq N-1; k \in \{1,2\}} \min_{\beta} \frac{(1-\beta R_i)^2 \|S(A_i,B)\|_F^2 + (1-\beta R_{i+k})^2 \|S(A_{i+k},B)\|_F^2}{\|S(A_i,B)\|_F^2 + \|S(A_{i+k},B)\|_F^2}. \tag{5.19}$$

Combining Eq. (5.19) and Lemma 5.5 gives

$$d^2(e) \geq \begin{cases} \frac{(2\delta^2)^{\log m} \delta^6 (1-\xi^2)^2}{4m\xi^4} & \delta^2 \leq 0.5 \\ \frac{\delta^8 (1-\xi^2)^2}{2m\xi^4} & \delta^2 > 0.5, \end{cases}$$

which concludes the proof of Lemma 5.4, and with it, Step 1 in the proof of Theorem 5.3. □

STEP 2: A SUFFICIENT CONDITION ON THE ESTIMATE $\hat{S}$.

Lemma 5.4 shows that there is a gap between the score of the correct placeholder edge and the scores of all other edges in $\mathscr{T}_1$. In the following lemma we show that if $\hat{S}$ is sufficiently close to S the gap is preserved and STDR selects the correct placeholder edge. For simplicity, we address only the case $\delta^2 > 0.5$.

LEMMA 5.7. Let $\mathscr{D} = \min\{\|S(A,B)\|_F, \|S(C_1,C_2)\|_F\}$. If the similarity matrix estimate $\hat{S}$ satisfies

$$\|S - \hat{S}\|_F \leq \frac{1}{2}\left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1+10\sqrt{2}}{\mathscr{D}^3}\right)^{-1} \frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2}, \tag{5.20}$$

then STDR selects the correct placeholder edge.

In our proof, we use the following auxiliary lemma, proven in Appendix F.

LEMMA 5.8. Let $d(e), \hat{d}(e)$ be the exact and estimated score functions. If $\|S - \hat{S}\|_F \leq \mathscr{D}/2$, then

$$|d(e) - \hat{d}(e)| \leq \|S - \hat{S}\|_F \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1+10\sqrt{2}}{\mathscr{D}^3}\right).$$

Proof of Lemma 5.7. Suppose $e^* \in \mathscr{T}_1$ is the correct placeholder edge and $e' \neq e^*$ is a different edge in $\mathscr{T}_1$. By Lemma 5.4

$$d(e') \geq \frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2},$$

while for the correct edge $d(e^*) = 0$. It follows from the triangle inequality that if

$$|d(e) - \hat{d}(e)| \leq \frac{1}{2}\frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2}, \tag{5.21}$$

for all edges e, then $\hat{d}(e^*) < \hat{d}(e')$. Since $\delta \leq \xi < 1$ and $m \geq 2$,

$$\frac{1}{2}\left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1+10\sqrt{2}}{\mathscr{D}^3}\right)^{-1} \frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2} \leq \frac{\mathscr{D}}{2}.$$

Thus, if the estimate $\hat{S}$ satisfies Eq. (5.20), then $\|S - \hat{S}\| \leq \mathscr{D}/2$ and the condition for Lemma 5.8 holds. Combining the lemma with Eq. (5.21) concludes the proof. □

STEP 3: FINITE SAMPLE GUARANTEES We are now ready to prove Theorem 5.3, which bounds the number of samples required to compute, with high probability, a sufficiently accurate estimate $\hat{S}$, as determined in Lemma 5.7.

Proof of Theorem 5.3. The following concentration bound for $\hat{S}$ was derived in Lemma 4.7 of [26],

$$\Pr(\|\hat{S} - S\|_F \leq t) \geq 1 - 2m^2 \exp\left(-\frac{2nt^2}{\ell^2 m^2}\right). \tag{5.22}$$

We note that in [26], this bound was presented for the spectral norm, but the proof holds for the Frobenius norm as well. Suppose that $\Pr(\|\hat{S} - S\|_F \leq t) > 1 - \varepsilon$, Namely

$$n \geq \frac{\ell^2 m^2}{2t^2} \log\left(\frac{2m^2}{\varepsilon}\right). \tag{5.23}$$

By Lemma 5.7, a sufficient condition for STDR to select the correct placeholder edge is

$$\|S - \hat{S}\|_F \leq \frac{1}{2}\left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1 + 10\sqrt{2}}{\mathscr{D}^3}\right)^{-1} \frac{\delta^4(1-\xi^2)}{\sqrt{2m}\xi^2}. \tag{5.24}$$

Setting t to the right-hand side of Eq. (5.24) and substituting into Eq. (5.23), we have that if

$$\begin{aligned} n &\geq \frac{\ell^2 m^2}{2} 2^2 \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1 + 10\sqrt{2}}{\mathscr{D}^3}\right)^2 \left(\frac{2m\xi^4}{\delta^8(1-\xi^2)^2}\right) \log\left(\frac{2m^2}{\varepsilon}\right) \\ &= 8\ell^2 m^3 \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1 + 10\sqrt{2}}{\mathscr{D}^3}\right)^2 \left(\frac{\xi^4}{\delta^8(1-\xi^2)^2}\right) \log\left(\frac{2m^2}{\varepsilon}\right) \end{aligned}$$

then Eq. (5.24) holds with probability at least $1-\varepsilon$, and thus the merging step in STDR selects the correct placeholder edge with high probability. □

5.3 *Guarantees for multiple partitions and merging steps*

The guarantees in Theorems 5.2 and 5.3 are derived for a single iteration of partitioning and merging. The following theorem, proved in the appendix, extends these theorems for multiple iterations.

THEOREM 5.4. Let $\mathscr{T}$ be a tree with m terminal nodes generated according to the molecular clock model. Let τ be the user defined parameter of the minimum partition. We assume that the provided algorithm Alg (see Algorithm 1) recovers trees with size up to τ exactly. The number of samples required to obtain an accurate tree recovery via STDR is of the order of

$$n = \tilde{\mathscr{O}}\left(\exp(2r(\mathscr{T}))m^2\left((1+\eta)^3 + m\min\left\{1, \frac{(1+\eta)^3}{\tau^2}\right\}\right)\right). \tag{5.25}$$

This analysis has important implications on the choice of the smallest partition τ in Algorithm 1. For balanced trees where $\eta = \mathscr{O}(1)$, the required number of samples is $\widetilde{O}(m^2)$ if τ is at least $O(\sqrt{m})$, but is $\widetilde{O}(m^3)$ if $\tau = \mathscr{O}(1)$. Thus on the one hand, reducing τ results in smaller subsets of terminal nodes, which improves the runtime of the reconstruction step of STDR. On the other hand it may affect the accuracy of the merging step. Figure 7 shows both runtime and accuracy of STDR as a function of the threshold parameter τ, when applying STDR with RAxML or SNJ as its subroutine. The data consists of $n = 1000$ samples generated from a binary symmetric tree with $m = 2048$ terminal nodes. The accuracy of the algorithm degrades for small values of τ while the runtime improves by approximately half an order of magnitude.

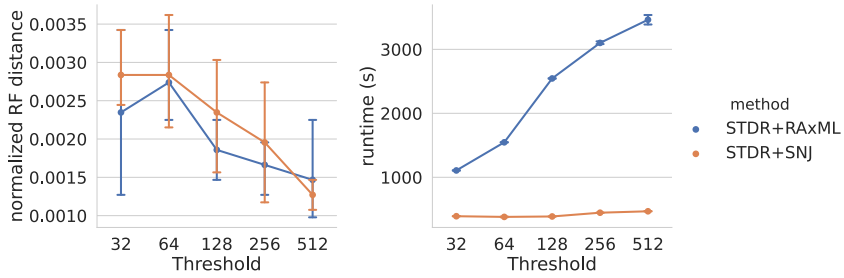


FIG. 7. Effect of minimal tree size τ on runtime and accuracy of SDTR. Various values of threshold τ were chosen to test the performance of SDTR method in recovering a binary tree of size 2048 from sequences of length 1000. SNJ and RAxML were used as the sub method of SDTR.

we compare this result to analogous results for three other methods that recover trees from sequences. For the comparison, we assume that τ is at least $\sqrt{m}$. For NJ, the sample complexity given in Section 3.3 of [5] is $\tilde{\mathcal{O}}(\exp(-4\min_{i,j}\ln(S(x_i,x_j))))$ or equivalently $\tilde{\mathcal{O}}\left(\delta^{-4r(\mathcal{T})}\right)$, For a binary symmetric tree $\mathrm{diam}(\mathcal{T}) = 2\log_2(m)$ and hence the complexity is $O(m^{8\log_2(1/\delta)})$, which is better than (5.25) for δ close to one, but worse for lower values of δ.

For SNJ, if $\delta^2 > 0.5$ the sample complexity is $\tilde{\mathcal{O}}(m^2)$ (by Theorem 4.3 in [26]). This improves upon Eq. (5.25) due to the additional factor of $\exp(-r(\mathcal{T}))$. For the Dyadic Closure method [15, Theorem 9), the sample complexity is $\tilde{\mathcal{O}}((1/\delta)^{4h(\mathcal{T})})$, where recall that $h(\mathcal{T})$ denotes the depth of a tree as in definition 5.1. For a binary symmetric tree $h(\mathcal{T}) = \log_2(m)$ in which case the complexity is $\mathcal{O}(m^{4\log_2(1/\delta)})$, which improves upon Eq. (5.25) by m^2. For highly imbalanced trees ($\mathrm{depth}(\mathcal{T}) = \mathcal{O}(1)$) the sample complexity is logarithmic in m. The improved sample complexity, however, comes at cost of a $\mathcal{O}(m^5)$ computational complexity.

6. Computational complexity

In this section, we analyze the computational complexity of STDR for trees satisfying the CBM model described above. First, we analyze the complexity of a single partitioning and merging iteration assuming that the partitions satisfy some balancedness constraint. Next, we derive the complexity for recovering a tree via STDR. Finally, we derive the complexity for the worst-case scenario of highly imbalanced partitions.

PARTITIONING Given the similarity matrix, partitioning a set of k terminal nodes requires $\mathcal{O}(k^2)$ operations, due to the computation of the Fiedler vector of the positive semi-definite Laplacian matrix. Computing two eigenvectors of an $k \times k$ matrix requires $\mathcal{O}(k^2)$ operations [53, Chapter 2).

MERGING Here we analyze the computational complexity of merging two subtrees. For ease of notations we assume that both subtrees are of equal size k. The same analysis applies for two subtrees that have less than k nodes. In Lemma 6.1, proved in Appendix J, we show that for two CBM trees with k nodes and η (defined in Eq. (5.3)) independent of k, the computational complexity of the merging step is of order $\mathcal{O}(k^2\log k)$.

LEMMA 6.1. Let $\mathcal{T}_1, \mathcal{T}_2$ be two trees with k terminal nodes that satisfy the assumptions of the CBM model. We assume that for both trees, the parameter η is upper bounded by some constant η_0, independent of k. Then, the computational complexity of merging trees $\mathcal{T}_1$ and $\mathcal{T}_2$ is $\mathcal{O}(k^2\log k)$.

REMARK 6.1. Lemma 6.1 can easily be extended for cases where the CBM model assumptions are not satisfied, as long as all partitions are balanced such that the ratio between the two parts is lower bounded by some constant.

Next, we use Lemma 6.1 to derive the complexity of multiple merging and partitioning iterations of STDR, denoted $T(m)$. We note that $T(m)$ does not include the complexity of the subroutine algorithm that recovers the structure of small trees. Each partitioning step for a CBM tree with m terminal nodes divides its terminal nodes into sets of sizes αm and $(1-\alpha)m$, where we assume that $\alpha_0 \leq \alpha \leq 0.5$, where $\alpha_0 = 1/(1+\eta)$. Thus, we have that

$$T(m) \leq \max_{\alpha_0 \leq \alpha < 0.5} \underbrace{\mathcal{O}(m^2)}_{\text{partitioning}} + T(\alpha m) + T((1-\alpha)m) + \underbrace{\mathcal{O}(m^2 \log m)}_{\text{merging}} \tag{6.1}$$

$$= \max_{\alpha_0 \leq \alpha < 0.5} T(\alpha m) + T((1-\alpha)m) + \mathcal{O}(m^2 \log m). \tag{6.2}$$

In Lemma J.2, we prove that if $T(m)$ satisfies the recursive relation in Eq. (6.1), then

$$T(m) = \mathcal{O}(m^2 \log m). \tag{6.3}$$

We denote by $\mathcal{B}(k)$ the complexity of recovering the topology of a tree with k terminal nodes by the given subroutine Alg. Thus, the total complexity of STDR including the recovery of small trees is

$$\mathcal{O}(m^2 \log m + (m/\tau)\mathcal{B}(\tau)).$$

Obviously, choosing different subroutines Alg. with different computational complexity will result in different total computational complexity, and might require different choice of τ. For example, the complexity of NJ is $\mathcal{B}(\tau) = \mathcal{O}(\tau^3)$. Thus, assuming the partitions satisfy the balancedness condition, the complexity of STDR + NJ is $\mathcal{O}(m^2 \log m + m\tau^2)$, which for $\tau = \mathcal{O}(1)$ improves upon the $\mathcal{O}(m^3)$ complexity of running NJ to recover the full tree. In the simulation section, we show that STDR + NJ outperforms NJ in accuracy while being about an order of magnitude faster.

COMPUTATIONAL ANALYSIS IN THE CASE OF HIGHLY IMBALANCED PARTITIONS. The worst-case scenario is the case of a highly imbalanced tree, such as the caterpillar tree. In this case, the cost of a single merging iteration of two trees with k terminal nodes each is $\mathcal{O}(k^3)$. Thus, in such cases there is no theoretical advantage over NJ in terms of computational complexity.

REMARK 6.2. An important property of the STDR algorithm, in terns of actual runtime, is that it is embarrassingly parallel. Specifically, steps 7 and 8 in Algorithm 1 can be executed in two independent processes. This may result in up to k parallel processes, where k is the number of partitions.

7. Simulation Results

We illustrate the performance of STDR in comparison to several other algorithms in a variety of simulated settings. To this end, we generated trees according to the birth death model, and the coalescent model.

In addition, we considered the challenging scenario of the caterpillar tree. In all experiments, the sequences were generated according to the HKY substitution model [24] with transition-transversion

ratio of 2, a typical value in the human genome [31]. The mutation rate for the HKY model is specified for each simulation. The trees were generated with the Dendropy python package [56]. Code for reproducing the results is available at https://github.com/aizeny/stdr.

We considered the following reconstruction methods: (i) RAxML [51], a standard tool for maximum likelihood-based tree inference, (ii) neighbour joining (NJ) and (iii) spectral neighbour joining (SNJ). Recall that STDR requires as input a subroutine *alg* for the reconstruction of the small trees. Thus, for comparison, we applied STDR with each of the aforementioned algorithms as the subroutine. We denote these three methods as (iv) STDR + RAxML, (v) STR + NJ and (vi) STDR + SNJ. A second input to STDR is the threshold parameter τ, which sets an upper bound for the size of the small trees. This parameter is specified in the description of each experiment. The accuracy of the different algorithms is measured by the normalized Robinson-Foulds (RF) distance, defined as the RF distance [17] between the reconstructed and reference tree divided by $2m - 6$. Each experiment was repeated five times to obtain a mean and standard deviation of the performance and runtime of each method.

In addition to the above experiments, we compare our merging procedure to TreeMerge [39]. The results for the caterpillar tree and the comparison to TreeMerge are shown in Appendix K. Finally, for a symmetric binary tree, we demonstrate how changes in the threshold τ affect the results of STDR.

Implementation remarks To improve the results of STDR, we computed two possible partitions C_1, C_2: (i) A partition that corresponds to a threshold at 0 in the Fiedler vector, and (ii) a partition that corresponds to the largest gap. In practice, the partition was chosen by method (i) or (ii), as the one that minimizes the second singular value of $S(C_1, C_2)$, see Lemma 3.1. To improve runtime, we apply randomized methods for computing leading singular values and vectors, see [1, 22, 53].

7.1 *Comparison of accuracy and runtime*

We generated random binary trees with $m = 2048$ terminal nodes according to the birth-death model [52]. We set a birth rate of 0.5 and zero death rate, with mutation rate equal to 0.05. The STDR threshold was set to $\tau = 256$ for all three methods. Figure 6 shows the accuracy and runtime of the different methods as a function of the sequence length n. Using STDR with NJ clearly improves upon the performance of standard NJ both in terms of accuracy and runtime. Compared to SNJ and RAxML, STDR + SNJ and STDR + RAxML show similar accuracy but with significantly faster runtimes. Similar results for trees generated according to the coalescent model and the caterpillar model appear in the appendix.

7.2 *Effect of threshold parameter*

Our aim in this experiment was to test the impact of the threshold parameter τ on the performance of STDR. To that end, we created a binary symmetric tree with $m = 2048$ terminal nodes and similarity between all adjacent nodes equal to $\delta = 0.65$. The number of samples was set to $n = 1000$. We then reconstructed the tree via STDR with different subroutines and a range of threshold values.

Figure 7 shows the normalized RF distance between the recovered trees and the ground truth tree as a function of the threshold. For both RAxML and SNJ, accuracy slightly improves for higher values of the threshold. STDR + NJ is not shown in the plot because it is significantly less accurate in this setting. These results are in accordance with our analysis in Section 5, where we show that the task of merging trees becomes challenging for small subsets of terminal nodes.

Acknowledgments

The authors would like to thank Junhyong Kim, Stefan Steinerberger and Ronald Coifman for useful and insightful discussions.

Funding

Y.K. and Y.A. acknowledge support by National Institutes of Health (NIH) (grant nos. R01GM131642, UM1DA051410 and R61DA047037). Y.K. and B.N. acknowledge support by NIH (grant R01GM135928). Y.K. acknowledges support by NIH (grant 2P50CA121974). B.N. acknowledges support by Isreal Science Foundation (ISF) (grant 2362/22).

Data Availability Statement

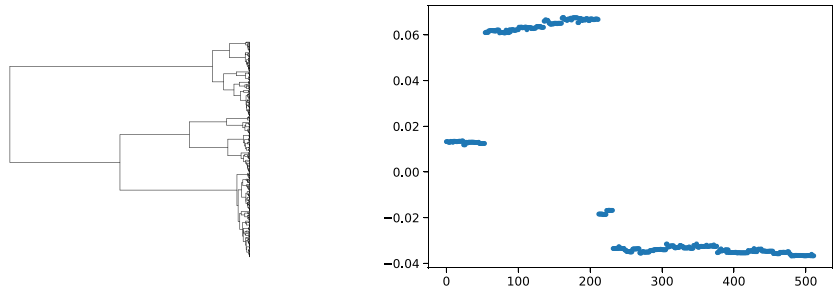
No new data were generated or analyzed in support of this review.

References

1. Aizenbud, Y. & Averbuch, A. (2019) Matrix decompositions using sub-Gaussian random matrices. *Inform. Inference: J. IMA*, **8**, 445–469.
2. Akra, M. & Bazzi, L. (1998) On the solution of linear recurrence equations. *Comput. Optim. Appl.*, **10**, 195–210.
3. Allman, E. S. & Rhodes, J. A. (2007) Molecular phylogenetics from an algebraic viewpoint. *Statist. Sinica*, **17**, 1299–1316.
4. Anandkumar, A., Hsu, D. J., Huang, F. & Kakade, S. M. (2012) Learning mixtures of tree graphical models. *Adv. Neural Inform. Process. Syst.*, **25**, 1052–1060.
5. Atteson, K. (1999) The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, **25**, 251–278.
6. Balakrishnan, S., Xu, M., Krishnamurthy, A. & Singh, A. (2011) Noise thresholds for spectral clustering. *Adv. Neural Inform. Process. Syst.*, **24**, 954–962.
7. Barriel, V. & Tassy, P. (1998) Rooting with multiple outgroups: consensus versus parsimony. *Cladistics*, **14**, 193–200.
8. Boykin, L. M., Kubatko, L. S. & Lowrey, T. K. (2010) Comparison of methods for rooting phylogenetic trees: a case study using Orcuttieae (Poaceae: Chloridoideae). *Mol. Phylogenet. Evol.*, **54**, 687–700.
9. Chang, J. T. (1996) Full reconstruction of Markov models on evolutionary trees: identifiability and consistency. *Math. Biosci.*, **137**, 51–73.
10. Choi, M. J., Tan, V. Y., Anandkumar, A. & Willsky, A. S. (2011) Learning latent tree graphical models. *J. Mach. Learn. Res.*, **12**, 1771–1812.
11. Crabtree, D. E. (1966) Applications of *M*-matrices to non-negative matrices. *Duke Math. J.*, **33**, 197–208.
12. Ding, C. H., He, X., Zha, H., Gu, M. & Simon, H. D. (2001) A min-max cut algorithm for graph partitioning and data clustering. *Proceedings 2001 IEEE International Conference on Data Mining*. NYC: IEEE, pp. 107–114.
13. Dorfler, F. & Bullo, F. (2012) Kron reduction of graphs with applications to electrical networks. *IEEE Trans. Circuits Syst. I: Regul. Pap.*, **60**, 150–163.
14. Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, England: Cambridge University Press.
15. Erdős, P. L., Steel, M. A., Székely, L. A. & Warnow, T. J. (1999) A few logs suffice to build (almost) all trees (I). *Random Struct. Algorithms*, **14**, 153–184.
16. Eriksson, N. (2005) Tree construction using singular value decomposition. *Algebraic Statistics for Computational Biology*. (L. Pachter & B. Sturmfels eds). Cambridge, England: Cambridge University Press, pp. 347–358.

17. Estabrook, G. F., McMorris, F. & Meacham, C. A. (1985) Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Zool.*, **34**, 193–200.
18. Felsenstein, J. (2003) *Inferring Phylogenies*, vol. **2**. Sunderland Massachusetts, USA: Sinauer Associates.
19. Fiedler, M. (1975) A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Math. J.*, **25**, 619–633.
20. Griffing, A. (2012) *Connections Between Numerical Taxonomy and Phylogenetics*. Ph.D Thesis. North Carolina State University.
21. Guindon, S. & Gascuel, O. (2003) A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst. Biol.*, **52**, 696–704.
22. Halko, N., Martinsson, P.-G. & Tropp, J. A. (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, **53**, 217–288.
23. Harmeling, S. & Williams, C. K. (2010) Greedy learning of binary latent trees. *IEEE Trans. Patt. Anal. Mach. Intell.*, **33**, 1087–1097.
24. Hasegawa, M., Kishino, H. & Yano, T.-A. (1985) Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J. Mol. Evol.*, **22**, 160–174.
25. Hillis, D. M., Moritz, C., Mable, B. K. & Olmstead, R. G. (1996) *Molecular Systematics*, vol. **1996**. MA: Sinauer Associates Sunderland, p. 1058.
26. Jaffe, A., Amsel, N., Aizenbud, Y., Nadler, B., Chang, J. T. & Kluger, Y. (2021) Spectral neighbor joining for reconstruction of latent tree models. *SIAM J. Math. Data Sci.*, **3**, 113–141.
27. Jaffe, A., Kluger, Y., Lindenbaum, O., Patsenker, J., Peterfreund, E. & Steinerberger, S. (2020) The spectral underpinning of word2vec. *Front. Appl. Math. Stat.*, **6**, 64.
28. Jiang, T., Kearney, P. & Li, M. (2001) A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. *SIAM J. Comput.*, **30**, 1942–1961.
29. Jones, M. G., Khodaverdian, A., Quinn, J. J., Chan, M. M., Hussmann, J. A., Wang, R., Xu, C., Weissman, J. S. & Yosef, N. (2020) Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. *Genome Biol.*, **21**, 1–27.
30. Jones, N. S. & Moriarty, J. (2013) Evolutionary inference for function-valued traits: Gaussian process regression on phylogenies. *J. R. Soc. Interface*, **10**, 20120616.
31. Keller, I., Bensasson, D. & Nichols, R. A. (2007) Transition-transversion bias is not universal: a counter example from grasshopper pseudogenes. *PLoS Genet*, **3**, e22.
32. Kinene, T., Wainaina, J., Maina, S. & Boykin, L. (2016) Rooting trees, methods for. *Encyclopedia Evol. Biol.*, **3**, 489–493.
33. Kingman, J. F. C. (1982) The coalescent. *Stochastic Process. Appl.*, **13**, 235–248.
34. Kumar, S. (2005) Molecular clocks: four decades of evolution. *Nat. Rev. Genet.*, **6**, 654–662.
35. Matsui, M. & Iwasaki, W. (2020) Graph splitting: a graph-based approach for superfamily-scale phylogenetic tree reconstruction. *Syst. Biol.*, **69**, 265–279.
36. Mihaescu, R., Levy, D. & Pachter, L. (2009) Why neighbor-joining works. *Algorithmica*, **54**, 1–24.
37. Mirarab, S. & Warnow, T. (2015) ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. *Bioinformatics*, **31**, i44–i52.
38. Molloy, E. K. & Warnow, T. (2019a) Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge. *Algorithms Mol. Biol.*, **14**, 14.
39. Molloy, E. K. & Warnow, T. (2019b) TreeMerge: a new method for improving the scalability of species tree estimation methods. *Bioinformatics*, **35**, i417–i426.
40. Morel, B., Barbera, P., Czech, L., Bettisworth, B., Hübner, L., Lutteropp, S., Serdari, D., Kostaki, E.-G., Mamais, I., Kozlov, A. M., et al. (2021) Phylogenetic analysis of SARS-CoV-2 data is difficult. *Mol. Biol. Evol.*, **38**, 1777–1791.
41. Mossel, E. & Roch, S. (2005) Learning nonsingular phylogenies and hidden Markov models. *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, 366–375.
42. Mourad, R., Sinoquet, C., Zhang, N. L., Liu, T. & Leray, P. (2013) A survey on latent tree models and applications. *J. Artif. Intell. Res.*, **47**, 157–203.

43. Nei, M. & Kumar, S. (2000) *Molecular Evolution and Phylogenetics*. Oxford, England: Oxford University Press.
44. Price, M. N., Dehal, P. S. & Arkin, A. P. (2010) FastTree 2-approximately maximum-likelihood trees for large alignments. *PLoS ONE*, **5**, e9490.
45. Quinn, J. J., Jones, M. G., Okimoto, R. A., Nanjo, S., Chan, M. M., Yosef, N., Bivona, T. G. & Weissman, J. S. (2021) Single-cell lineages reveal the rates, routes, and drivers of metastasis in cancer xenografts. *Science*, **371**, aseabc1944.
46. Saitou, N. & Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.
47. Sanderson, M. J. & Driskell, A. C. (2003) The challenge of constructing large phylogenetic trees. *Trends Plant Sci.*, **8**, 374–379.
48. Semple, C., Steel, M., et al. (2003) *Phylogenetics*, vol. **24**. Oxford, England: Oxford University Press on Demand.
49. Simeonov, K. P., Byrns, C. N., Clark, M. L., Norgard, R. J., Martin, B., Stanger, B. Z., Shendure, J., McKenna, A. & Lengner, C. J. (2021) Single-cell lineage tracing of metastatic cancer reveals selection of hybrid EMT states. *Cancer Cell.*, **39**, 1150–1162.e9.
50. Sokal, R. R. (1958) A statistical method for evaluating systematic relationships. *Univ. Kansas. Sci. Bull.*, **38**, 1409–1438.
51. Stamatakis, A. (2014) RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, **30**, 1312–1313.
52. Steel, M. (2016) *Phylogeny: Discrete and Random Processes in Evolution*. Philadelphia, USA: SIAM.
53. Stewart, G. W. (2001) *Matrix Algorithms: vol. II: Eigensystems*. Philadelphia, USA: SIAM.
54. Stone, E. A. & Griffing, A. R. (2009) On the Fiedler vectors of graphs that arise from trees by Schur complementation of the Laplacian. *Linear Algebra Appl.*, **431**, 1869–1880.
55. Strimmer, K. & Von Haeseler, A. (1996) Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. *Mol. Biol. Evol.*, **13**, 964–969.
56. Sukumaran, J. & Holder, M. T. (2010) DendroPy: a python library for phylogenetic computing. *Bioinformatics*, **26**, 1569–1571.
57. Tamura, K., Nei, M. & Kumar, S. (2004) Prospects for inferring very large phylogenies by using the neighbor-joining method. *Proc. Natl. Acad. Sci.*, **101**, 11030–11035.
58. Von Luxburg, U. (2007) A tutorial on spectral clustering. *Statist. Comput.*, **17**, 395–416.
59. Wilkinson, M., McInerney, J. O., Hirt, R. P., Foster, P. G. & Embley, T. M. (2007) Of clades and clans: terms for phylogenetic relationships in unrooted trees. *Trends Ecol. Evol.*, **22**, 114–115.
60. Warnow, T. (2018) Supertree construction: opportunities and challenges. arXiv preprint arXiv:1805.03530.
61. Yang, Z. & Rannala, B. (2012) Molecular phylogenetics: principles and practice. *Nat. Rev. Genet.*, **13**, 303.
62. Yu, Y., Wang, T. & Samworth, R. J. (2015) A useful variant of the Davis–Kahan theorem for statisticians. *Biometrika*, **102**, 315–323.
63. Zhang, N. L., Yuan, S., Chen, T. & Wang, Y. (2008) Latent tree models and diagnosis in traditional Chinese medicine. *Artif. Intell. Med.*, **42**, 229–245.
64. Zhang, S.-B., Zhou, S.-Y., He, J.-G. & Lai, J.-H. (2011) Phylogeny inference based on spectral graph clustering. *J. Comput. Biol.*, **18**, 627–637.
65. Zhou, X., Shen, X.-X., Hittinger, C. T. & Rokas, A. (2018) Evaluating fast maximum likelihood-based phylogenetic programs using empirical phylogenomic data sets. *Mol. Biol. Evol.*, **35**, 486–503.



(a) Generated coalescent tree (b) Fiedler vector of the coalescent tree

FIG. A1. Coalescent tree example with 512 terminal nodes

A. Example of Fiedler vector in a coalescent tree

We generated a tree with $m = 512$ nodes according to the coalescent model, see Figure A1(a). The transition matrices were set according to the HKY model [24]. We then generated a dataset of nucleotide sequences of length $n = 2000$. Figure A1(b) shows the Fiedler vector of the similarity graph estimated from the dataset. Partitioning the terminal nodes according to the sign pattern of the Fiedler vector yields two clans.

B. Relation between the partitioning step and the min-cut criterion

Let $\mathscr{T}$ be a binary tree and G be its similarity graph, as defined in Section 3.1. The following lemma shows that partitioning the terminal nodes according to the min-cut criterion yields two clans of $\mathscr{T}$.

LEMMA B.1. Let G be the similarity graph of a binary tree $\mathscr{T}$. Let A^* and B^* be a partition of the terminal nodes that minimizes the following min-cut criterion:

$$(A^*, B^*) \in \operatorname*{argmin}_{A,B} \mathrm{Cut}_G(A, B) = \operatorname*{argmin}_{A,B} \sum_{i \in A, j \in B} S(x_i, y_j). \tag{B.1}$$

Then A^* and B^* are clans in $\mathscr{T}$.

Proof. Let (x_1, x_2) be a pair of adjacent terminal nodes. Consider an arbitrary partition of the terminal nodes into two non-empty subsets, denoted A and B. The two adjacent nodes (x_1, x_2) can, respectively, be labeled (A, B), (A, A), (B, A) or (B, B). We show that if A and B each contains nodes besides x_1 and x_2, then assigning x_1 and x_2 to the same subset decreases the value of the min-cut criterion.

Assume without loss of generality that $x_1 \in A, x_2 \in B$. The cut between A and B is equal to

$$\mathrm{Cut}(A, B) \equiv \sum_{x \in A, x' \in B} S(x, x') = S(x_1, x_2) + \sum_{x' \in B \setminus \{x_2\}} S(x_1, x') + \sum_{x \in A \setminus \{x_1\}} S(x, x_2) + S_0,$$

where

$$S_0 = \sum_{\substack{x\in A\setminus\{x_1\}\\ x'\in B\setminus\{x_2\}}} S(x,x')$$

does not depend on the assignment of x_1 and x_2. Let h be the unique node that is adjacent to both x_1 and x_2. From the multiplicative property of the similarity, we have

$$\mathrm{Cut}(A,B) = S(x_1,x_2) + S(x_1,h)\sum_{x'\in B\setminus\{x_2\}} S(h,x') + S(x_2,h)\sum_{x\in A\setminus\{x_1\}} S(x,h) + S_0.$$

Without loss of generality, assume that

$$\sum_{x'\in B\setminus\{x_2\}} S(h,x') \geq \sum_{x\in A\setminus\{x_1\}} S(x,h). \tag{B.2}$$

It follows that

$$\begin{aligned}\mathrm{Cut}(A,B) &\geq S(x_1,h)\sum_{x\in A\setminus\{x_1\}} S(x,h) + S(x_2,h)\sum_{x\in A\setminus\{x_1\}} S(x,h) + S_0 \qquad (B.3)\\ &= \sum_{x\in A\setminus\{x_1\}} S(x,x_1) + \sum_{x\in A\setminus\{x_1\}} S(x,x_2) + \sum_{\substack{x\in A\setminus\{x_1\}\\ x'\in B\setminus\{x_2\}}} S(x,x') = \sum_{\substack{x\in A\setminus\{x_1\}\\ x'\in B\cup\{x_1\}}} S(x,x').\end{aligned}$$

Note that the right hand side of Eq. (B.3) equals the value of the cut of the same partition, but with x_1 moved from A to B. Thus, the min-cut partition $\{A^*, B^*\}$ satisfies one of the following:

- x_1 and x_2 are in the same subset.
- One of A^* or B^* equals exactly to $\{x_1\}$ or $\{x_2\}$.

Next, let C_1 and C_2 be two adjacent clans. Assume that the terminal nodes of each of the clans are homogeneous (i.e., they all belong to the same subset, A or B). The same argument for a pair of terminal nodes carries over to the case of two adjacent homogeneous clans, showing that the minimal cut partition $\{A^*, B^*\}$ satisfies one of the following:

- C_1 and C_2 are in the same subset.
- One of A^* or B^* equals exactly C_1 or C_2.

Let $\{A, B\}$ be an arbitrary partition of the terminal nodes that does not correspond to two clans in the tree. Since A and B are not clans, there must be at least two disjoint pairs C_1, C_2 and $\tilde{C}_1, \tilde{C}_2$ of homogeneous adjacent subsets, where the nodes in C_1 are labeled by A and the nodes in C_2 are labeled by B. By our arguments $\mathrm{Cut}(A,B)$ can be reduced by either changing the labels of C_1 to B or C_2 to A which implies that $\{A, B\}$ is not the min-cut partition. Thus, for any min-cut partition $\{A^*, B^*\}$, A^* and B^* are clans. □

C. Supplementary proofs for Section 3

We present here the proofs of Lemmas 3.2 and Lemma 3.3 that are used in Section 3.

Proof of Lemma 3.2. Let C_2 be the clan of all the terminal nodes of $\mathscr{T}$ that are not in C_1. Consider an edge $e(h_A, h_B)$ in $\mathscr{T}_1$ that partitions C_1 into $A(e)$ and $B(e)$. First, assume that $e(h_A, h_B)$ is the correct placeholder edge of $\mathscr{T}_1$. Then there exists a node h_1 in the full tree $\mathscr{T}$ that is connected to h_A, h_B and to the root node of C_2. Removing the edge $e(h_A, h_1)$ in $\mathscr{T}$ separates the subset $A(e)$ from the remaining nodes in $\mathscr{T}$, which implies that $A(e)$ is a clan in $\mathscr{T}$. By the same argument, $B(e)$ is also a clan in $\mathscr{T}$.

Conversely, assume that $A(e)$, $B(e)$ and C_2 are disjoint clans that partition the terminal nodes of $\mathscr{T}$. Then, there exists a node h_1 that connects to the roots of $A(e), B(e)$ and $\mathscr{T}_2$. This proves that the edge $e(h_A, h_B)$ in $\mathscr{T}_1$ is the correct placeholder edge, since it is where the root h_1 is inserted. □

Proof of Lemma 3.3. Let $C_1 = A \cup B$ be the terminal nodes of the clan $\mathscr{T}_1$ and let h_1 be its root. We denote by C_2 the terminal nodes in its adjacent clan. By the multiplicative property of the similarity function,

$$S(C_1, C_2) = S(C_1, h_1)S(h_1, C_2).$$

Combining the above expression with Eq. (3.4) implies that the left singular vector u of $S(C_1, C_2)$ is proportional to the vector of similarities $S(C_1, h_1)$. That is, $\exists \beta \in \mathbb{R}$ such that $S(C_1, h_1) = \beta u$. Let e be an edge in $\mathscr{T}_1$ that partitions the terminal nodes into $A(e), B(e)$. The vector $S(C_1, h_1)$ can be similarly partitioned into $S(A(e), h_1)$ and $S(B(e), h_1)$ such that

$$S(A(e), h_1) = \beta u_{A(e)}, \qquad S(B(e), h_1) = \beta u_{B(e)}. \tag{C.1}$$

We first prove that if e is the correct placeholder edge of $\mathscr{T}_1$, then Eq. (3.5) holds. By Lemma 3.2, if e is the correct placeholder edge then the root node h_1 separates $A(e)$ from $B(e)$. By Eq. (C.1) and the multiplicative property of the similarity measure, we have

$$S(A(e), B(e)) = S(A(e), h_1)S(h_1, B(e)) = u_{A(e)}\beta^2 u_{B(e)}^T.$$

Setting $\alpha = \beta^2$ proves Eq. (3.5).

Next, we assume that Eq. (3.5) holds for some edge e and prove that e is the correct placeholder edge. Consider the matrix $S(A(e), B(e) \cup C_2)$. Since h_1 is the root of $\mathscr{T}_1$,

$$S(A(e), C_2) = S(A(e), h_1)S(h_1, C_2) \qquad \text{and} \qquad S(A(e), h_1) = \beta u_{A(e)}$$

we have

$$S(A(e), C_2) = \beta u_{A(e)} S(h_1, C_2).$$

Recall that by assumption $S(A(e), B(e)) = u_{A(e)}\alpha u_{B(e)}$. It follows that both matrices $S(A(e), B(e))$ and $S(A(e), C_2)$ are rank one with a left singular vector equal to $u_{A(e)}$. Thus, the concatenated matrix $S(A(e), B(e) \cup C_2)$ is rank-one. By Lemma 3.1, this implies that $A(e)$ is a clan of the tree $\mathscr{T}$. A similar argument shows that $B(e)$ is also a clan in $\mathscr{T}$. Since $A(e)$ and $B(e)$ are both clans in $\mathscr{T}$, it follows from Lemma 3.2 that e is the correct placeholder edge of $\mathscr{T}_1$. □

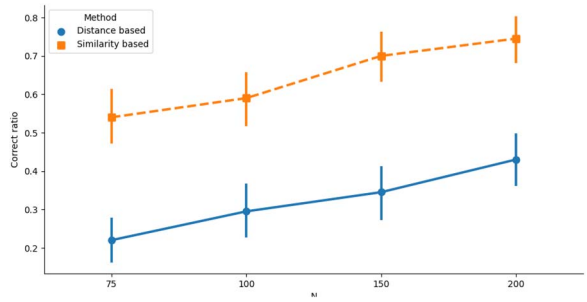


FIG. D2. Partitioning accuracy vs. number of samples. Comparison between distance based and similarity based partitioning.

D. Comparison to distance based tree partitioning

Let $D \in \mathbb{R}^{m\times m}$ be a matrix whose elements are the pairwise phylogenetic distances between all terminal nodes. Given the exact distance matrix, it was shown in [20] that the terminal nodes of a tree can be partitioned into two clans according to the sign pattern of the leading eigenvector of the following matrix

$$(I - \mathbf{1}\mathbf{1}^T/m)D(I - \mathbf{1}\mathbf{1}^T/m).$$

Figure D2 shows the percentage of times the terminal nodes were correctly partitioned into clans by applying our similarity based approached vs. the distance-based approach derived in [20]. We generated 200 random trees according to the birth-death model with $m = 128$ terminal nodes. The Figure shows the ratio of times each method successfully partitioned the tree as a function of the number of samples with a fixed mutation rate between adjacent nodes of $\delta = 0.9$. The advantage of using the similarity matrix over the distance matrix is clear.

E. Proof of Lemma 4.1

We begin with several definitions and notations. We denote by $G(v,w)$, $\mathscr{T}(v,w)$ the weight between nodes v and w in a graph G and tree $\mathscr{T}$, respectively. For a tree $\mathscr{T}$, we denote by $path_{\mathscr{T}}(v,w)$ the set of edges on the path between nodes v and w,

$$\text{path}_{\mathscr{T}}(v,w) = \{(\tilde{v},\tilde{w})|\ \tilde{v} \text{ and } \tilde{w} \text{ are adjacent nodes on the path between } v \text{ and } w\}.$$

Next, we define the multiplicative weight between two nodes in a tree.

DEFINITION E.1. The multiplicative weight between v and w in a tree $\mathscr{T}$ is equal to,

$$\alpha_{\mathscr{T}}(v,w) = \prod_{(\tilde{v},\tilde{w})\in \text{path}_{\mathscr{T}}(v,w)} \mathscr{T}(\tilde{v},\tilde{w}). \tag{E.1}$$

For example, let $\mathscr{T}$ be a tree whose edge weights are given by the similarity in Eq. (3.2), then the similarity between two terminal nodes x_1, x_2 is equal to the multiplicative weight $\alpha_{\mathscr{T}}(x_1,x_2)$. The next definition concerns a graph with nodes that correspond to a subset of nodes in $\mathscr{T}$, and weights computed according to (E.1).

DEFINITION E.2. Multiplicative subgraphLet $\mathscr{T}$ be a tree with a set of nodes V. We say that a graph G is a *multiplicative subgraph* with respect to $\mathscr{T}$ and a subset of nodes $\widetilde{V} \subset V$ if (i) the nodes of G correspond to $\widetilde{V}$ and (ii) the weight assigned to an edge connecting v, w in G is equal to the multiplicative weight between v and w in $\mathscr{T}$,

$$G(v, w) = \alpha_{\mathscr{T}}(v, w).$$

For convenience, we will sometimes say that G is a multiplicative subgraph of $\mathscr{T}$ without explicitly stating which nodes are included in G. By definition, the similarity graph G is a multiplicative subgraph with respect to the terminal nodes of $\mathscr{T}$. Note that we use v and w as nodes both in G and in $\mathscr{T}$ interchangeably, since by definition every node in G corresponds to a node in $\mathscr{T}$.

The proof of Lemma 4.1 is constructive. Given a tree $\mathscr{T}$ and its similarity graph G, we present an iterative procedure to build a second tree $\tilde{\mathscr{T}}$, with the same topology as $\mathscr{T}$, but with different weights such that

$$L_G = L_{\tilde{\mathscr{T}}/R},$$

where R is the set of all internal nodes in $\mathscr{T}$. Computing $\tilde{\mathscr{T}}$ consists of iterative and simultaneous updates of a graph and a tree: (i) a graph G_i with nodes that correspond to a subset of the nodes in $\mathscr{T}$. The initial graph G_0 is set to G, with only the terminal nodes of $\mathscr{T}$. (ii) A tree $\mathscr{T}_i$, with the same topology as $\mathscr{T}$. The weights of the initial tree $\mathscr{T}_0$ are set such that $\mathscr{T}_0 = \mathscr{T}$.

At each iteration i, we add one of the non-terminal nodes h_i of $\mathscr{T}$ (that was not previously added) to G_i, creating G_{i+1}. The weights of the new graph G_{i+1} are set such that the Schur complement of its Laplacian matrix with respect to the added node h_i is equal to the Laplacian of the previous graph L_{G_i}.

$$L_{G_i} = L_{G_{i+1}/h_i}. \tag{E.2}$$

The steps for computing G_{i+1} given G_i and $\mathscr{T}_i$ are described in Algorithm 2. Next, we compute a new tree $\mathscr{T}_{i+1}$ with the same topology as $\mathscr{T}_i$. The weights of $\mathscr{T}_{i+1}$ are set such that G_{i+1} becomes a multiplicative subgraph with respect to $\mathscr{T}_{i+1}$. The steps for computing $\mathscr{T}_{i+1}$ are described in Algorithm 2. At every iteration i, we maintain an *active set* of nodes which we denote by A_i. When updating G_i, changes are only made to edges connecting two nodes in $A_i \cup h_i$. When updating $\mathscr{T}_i$, changes are only made to edges on the path between two nodes in the active set. The initial active set A_0 is equal to all terminal nodes of $\mathscr{T}$.

In our proof, we use the following two auxiliary lemmas, that show the correctness of the updating procedure of G_i and $\mathscr{T}_i$. An implementation of Algorithms 2 and 3 is available on GitHub. The first lemma proves the correctness of Algorithm 2. The input to Algorithm 2 is the tree $\mathscr{T}_i$, a multiplicative subgraph G_i and an active set A_i, all of which were computed in the previous iteration. The output of the algorithm is an updated graph G_{i+1} that contains an additional node h_i. In addition, the algorithm updates the active set A_i and creates A_{i+1}.

LEMMA E.1. The output of Algorithm 2 is a graph G_{i+1} whose nodes include h_i as well as all the nodes in G_i such that

$$L_{G_{i+1}/h_i} = L_{G_i}.$$

The next lemma concerns the updating procedure of $\mathscr{T}_i$. The input to Algorithm 3 consists of the new active set A_{i+1}, and the node h_i added to G_{i+1}. Here, the only changes made are to edges on the path between h_i and the nodes in the active set A_{i+1}.

LEMMA E.2. The tree $\mathscr{T}_{i+1}$ built according to Algoithm 3 is such that G_{i+1} becomes a multiplicative subgraph of $\mathscr{T}_{i+1}$.

Figure E3 shows two iterations of the aforementioned process for a tree $\mathscr{T}$ with four terminal and two non-terminal nodes. For simplicity, all the weights of the tree $\mathscr{T}$ are set to $1/2$.

Algorithm 2 Updating G_i

Input: $\mathscr{T}_i$ - a tree graph
G_i - a multiplicative subgraph of $\mathscr{T}_i$
A_i - active set of nodes

Output: G_{i+1} - updated graph such that $L_{G_i} = L_{G_{i+1}/h_i}$
A_{i+1} - updated active set
h_i - the node added to G_i
$v_{i,1}, v_{i,2}$ - nodes removed form the active set

1: Initialize $G_{i+1} = G_i$ and $A_{i+1} = A_i$.
2: Choose a node h_i in $\mathscr{T}_i$ that is not in G_i and is adjacent to at least two nodes $v_{i,1}, v_{i,2}$ in the active set A_i. Add h_i to G_{i+1}.
3: Remove edges between the nodes $v_{i,1}, v_{i,2}$ and the rest of the active set A_i.
4: The weight between the new node h_i and a node x in the active set is computed by

$$G_{i+1}(h_i, x) = d\,\alpha_{\mathscr{T}_i}(h_i, x), \tag{E.3}$$

where

$$d = \sum_{x' \in A_i} \alpha_{\mathscr{T}_i}(h_i, x'). \tag{E.4}$$

5: The weights between two nodes x, y in the active set (except $v_{i,1}, v_{i,2}$) are updated by

$$G_{i+1}(x, y) = G_i(x, y) - \alpha_{\mathscr{T}_i}(h_i, x)\alpha_{\mathscr{T}_i}(h_i, y). \tag{E.5}$$

6: Remove the nodes $v_{i,1}$ and $v_{i,2}$ from the active set A_{i+1}, and add h_i.
7: **return** G_{i+1}, A_{i+1}, h_i,$v_{i,1}$,and $v_{i,2}$.

Proof of Lemma 4.1. We initialize the updating process with a tree $\mathscr{T}$ and its similarity matrix $G = G_0$. By definition, G_0 is a multiplicative subgraph of $\mathscr{T}$, and therefore satisfies the condition for Lemma E.1. The lemma guarantees that after the first update, we obtain a graph G_1 with a Laplacian that satisfies,

$$L_{G_0} = L_{G_1/h_0},$$

where h_0 is the node added to G_0 at the first iteration. Lemma E.2 guarantees that G_1 is a multiplicative subgraph of $\mathscr{T}_1$. Thus, we can re-apply Algorithm 2 with the pair $G_1, \mathscr{T}_1$. Thus, at each iteration i, we obtain a graph G_{i+1} that satisfies,

$$L_{G_i} = L_{G_{i+1}/h_i}. \tag{E.6}$$

Algorithm 3 Updating $\mathscr{T}_i$

Input: $\mathscr{T}_i$ - a tree graph
A_{i+1} - the active set
h_i - the node last added to G_{i+1}
$v_{i,1}, v_{i,2}$ - nodes that where removed from the active set in the last update
Output: $\mathscr{T}_{i+1}$ - a tree with weights computed such that G_{i+1} is a multiplicative subgraph of $\mathscr{T}_{i+1}$

1: Set $\mathscr{T}_{i+1}(h_i, v_{i,1}) = d\,\mathscr{T}_i(h_i, v_{i,1})$ and $\mathscr{T}_{i+1}(h_i, v_{i,2}) = d\,\mathscr{T}_i(h_i, v_{i,2})$
2: For node $x \notin \{v_{i,1}, v_{i,2}\}$ adjacent to h_i, set

$$\mathscr{T}_{i+1}(h_i, x) = \frac{d\,\mathscr{T}_i(h_i, x)}{\sqrt{1 - \alpha_{\mathscr{T}_i}(x, h_i)^2}},$$

where d is given by Eq. (E.4).
3: For two adjacent nodes $x, y \in \mathscr{T}$ where y is a node in the active set A_{i+1} and x is other path between y and h_i, set

$$\mathscr{T}_{i+1}(x, y) = \mathscr{T}_i(x, y)\sqrt{1 - \alpha_{\mathscr{T}_i}(x, h_i)^2}$$

4: For two adjacent nodes $x, y \in \mathscr{T}$ that are not in the active set. If $\mathscr{T}_i(x, y)$ is on the path between a node in the active set and h_i, where x is closer to h_i, set

$$\mathscr{T}_{i+1}(x, y) = \mathscr{T}_i(x, y)\frac{\sqrt{1 - \alpha_{\mathscr{T}_i}(x, h_i)^2}}{\sqrt{1 - \alpha_{\mathscr{T}_i}(y, h_i)^2}}$$

5: **return** $\mathscr{T}_{i+1}$

Repeating the updating process for all l non-terminal nodes of $\mathscr{T}$ yields the graph G_l, which by construction has the same topology as $\mathscr{T}$. In addition, due to the transitivity of the Schur's complement operation, Eq. (E.6) implies that

$$L_{\mathscr{T}_l/R} = L_{G_l/R} = L_{G_l/\{h_0,\ldots h_{l-1}\}} = L_{G_{l-1}/\{h_0,\ldots h_{l-2}\}} = \ldots = L_{G_1/h_0} = L_{G_0} = L_G.$$

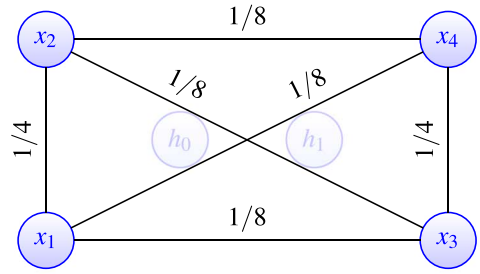
Thus, $\mathscr{T}_l$ is a tree with the same topology as $\mathscr{T}$, but with different weights such that $L_{\mathscr{T}_l/R} = L_G$, which proves the lemma. □

Proof of Lemma E.1. Assume, for simplicity of notation, that the jth row/column of L_G is the row/column that correspond to h_j for any j such that

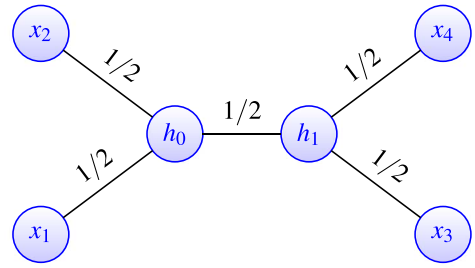
$$L_G(i, j) = -G(h_i, h_j) \qquad \forall h_i, h_j \in G \text{ with } i \neq j.$$

We denote by m_j the j-th column of $L_{G_{i+1}}$ after removing the i-th entry, and by $\mathbf{1}$ the all one vector. Since h_i is a single node, the Schur complement L_{G_{i+1}/h_i} defined in (4.3) can be simplified to

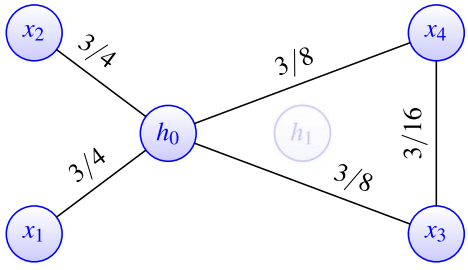
$$L_{G_{i+1}/h_i}(j, k) = L_{G_{i+1}}(j, k) - \frac{(\mathbf{1}^T m_j)(\mathbf{1}^T m_k)}{\sum_{l \neq i} \mathbf{1}^T m_l}. \tag{E.7}$$



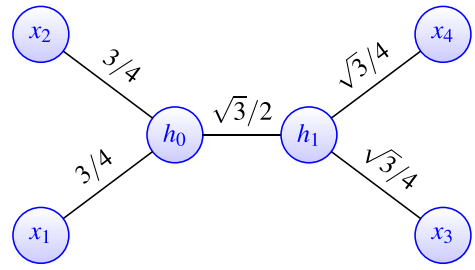
(a) A graph G_0 that is a multiplicative sub-graph with respect to $\mathscr{T}$ and the the terminal nodes of $\mathscr{T}$.



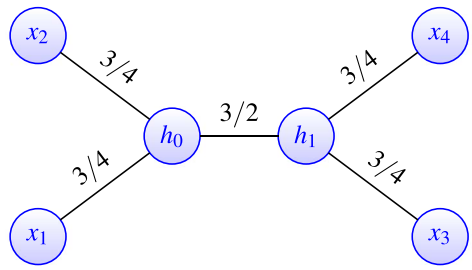
(b) A tree $\mathscr{T}_0$ with 4 terminal nodes and 2 internal nodes. The weight over all edges is equal to $1/2$.



(c) A graph G_1, created by adding the node h_0 to G_0. The weights of the graph are set such that $L_{G_0} = L_{G_1/h_0}$.



(d) A tree $\mathscr{T}_1$ with weights set such that G_1 is a multiplicative subgraph of $\mathscr{T}_1$ with respect to $\{x_1, x_2, x_3, x_4, h_1\}$



(e) A tree $\mathscr{T}_2$ with weights set such that $L_{\mathscr{T}_2/h_2} = L_{G_1}$.

FIG. E3. Constructing a tree $\mathscr{T}_2$ such that the Schur complement of its Laplacian with respect to the internal nodes is equal to L_G.

For a Laplacian matrix, the sum over any row is equal to zero. Since m_j is equal to the row of $L_{G_{i+1}}$ after removing the i-th entry we have that $\mathbf{1}^T m_j = -L_{G_{i+1}}(i,j)$. We rewrite Eq. (E.7) as,

$$L_{G_{i+1}/h_i}(j,k) = L_{G_{i+1}}(j,k) + \frac{L_{G_{i+1}}(j,i)L_{G_{i+1}}(k,i)}{\sum_{l\neq i} L_{G_{i+1}}(i,l)}. \tag{E.8}$$

The only edges changed between G_i and G_{i+1} are edges between nodes in the active set A_i. Thus, if either h_k or h_j are not in the active set then $L_{G_{i+1}}(j,k) = L_{G_i}(j,k)$. In addition, by step 4 of Algorithm

2, the added node h_i is only connected to nodes in the active set A_i. Thus, if either node h_k or h_j are not part of A_i we have $L_{G_{i+1}}(j,i)L_{G_{i+1}}(k,i) = 0$. It follows that in this case $L_{G_{i+1}/h_i}(j,k) = L_{G_i}(j,k)$ as required.

Next, we assume that both h_j and h_k are part of the active set A_i. Eqs. (E.3) and (E.5) give

$$L_{G_{i+1}}(j,k) = L_{G_i}(j,k) + \alpha_{\mathscr{T}_i}(h_i,h_j)\alpha_{\mathscr{T}_i}(h_i,h_k), \qquad L_{G_{i+1}}(k,i) = -d\alpha_{\mathscr{T}_i}(h_i,h_k). \tag{E.9}$$

By step 4 of Algorithm 2, h_i is only connected to nodes in the active set A_i. Inserting Eq. (E.9) to Eq. (E.8) gives

$$L_{G_{i+1}/h_i}(j,k) = L_{G_i}(j,k) + \alpha_{\mathscr{T}_i}(h_i,h_j)\alpha_{\mathscr{T}_i}(h_i,h_k) - \frac{d^2\alpha_{\mathscr{T}_i}(h_i,h_j)\alpha_{\mathscr{T}_i}(h_i,h_k)}{\sum_{x\in A_i} d\alpha_{\mathscr{T}_i}(h_i,x)}, \tag{E.10}$$

The denominator in the last term on the r.h.s of Eq. (E.10) is equal to d^2 and hence,

$$L_{G_{i+1}/h_i}(j,k) = L_{G_i}(j,k) + \alpha_{\mathscr{T}_i}(h_i,h_j)\alpha_{\mathscr{T}_i}(h_i,h_k) - d^2\alpha_{\mathscr{T}_i}(h_i,h_j)\alpha_{\mathscr{T}_i}(h_i,h_k)\frac{1}{d^2} = L_{G_i}(j,k).$$

We conclude that for any element j,k we have $L_{G_{i+1}/h_i}(j,k) = L_{G_i}(j,k)$. □

Proof of Lemma E.2. Here, our task is to prove that the weight assigned to any edge $G_{i+1}(x,y)$ is equal to the multiplicative path $\alpha_{\mathscr{T}_{i+1}}(x,y)$. We address three cases: (i) the node x is in the active set A_{i+1} and y is equal to the node h_i added to the graph in iteration i. (ii) Both x and y are in A_{i+1}, and are not equal to h_i, and (iii) $x = h_i$ and y is either $v_{i,1}$ or $v_{i,2}$. For a pair of nodes (x,y) that is not in $(i)-(iii)$ the edges in G_i and $\mathscr{T}_i$ were not changed in the updating steps.

For case (i) we assume that x is in A_{i+1} and $y = h_i$ and hence by Eq. (E.3) in Algorithm 2

$$G_{i+1}(x,h_i) = d\alpha_{\mathscr{T}_i}(x,h_i).$$

We denote the nodes on the path between x and h_i in $\mathscr{T}_i$ by

$$path(h_i,x) = \{z_1 = h_i, z_2, \ldots, z_K = x\}.$$

The edge between h_i and z_2 is updated according to step 2 of Algorithm 3. The edge between z_{K-1} and z_K is updated by step 3. The remaining edges are updated by step 4. The multiplicative weight $\alpha_{\mathscr{T}_{i+1}}(x,h_i)$

in the updated tree $\mathscr{T}_{i+1}$ according to Algorithm 3 is equal to

$$\begin{aligned}\alpha_{\mathscr{T}_{i+1}}(x,h_i) &= \prod_{j=1}^{K-1}\mathscr{T}_{i+1}(z_j,z_{j+1})\\ &= \frac{d\mathscr{T}_i(h_i,z_2)}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_2,h_i)^2}}\times\\ &\prod_{j=2}^{K-2}\mathscr{T}_i(z_j,z_{j+1})\frac{\sqrt{1-\alpha_{\mathscr{T}_i}(z_j,h_i)^2}}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_{j+1},h_i)^2}}\sqrt{1-\alpha_{\mathscr{T}_i}(z_{K-1},h_i)^2}\mathscr{T}_i(z_{K-1},x)\\ &= d\prod_{j=1}^{K-1}\mathscr{T}_i(z_j,z_{j+1}) = d\alpha_{\mathscr{T}_i}(x,h_i). \end{aligned} \tag{E.11}$$

Thus, the weight $G_{i+1}(x,h_i)=\alpha_{\mathscr{T}_{i+1}}(x,h_i)$ for any x in the active set.

In case (ii) x,y are two nodes in the active set not equal to h_i. According to Eq. (E.5) in Algorithm 2

$$G_{i+1}(x,y) = G_i(x,y) - \alpha_{\mathscr{T}_i}(h_i,x)\alpha_{\mathscr{T}_i}(h_i,y).$$

Denote by u the unique node that connects between the nodes x,y and h_i. Then,

$$\alpha_{\mathscr{T}_i}(h_i,x)\alpha_{\mathscr{T}_i}(h_i,y) = \alpha_{\mathscr{T}_i}(h_i,u)^2\alpha_{\mathscr{T}_i}(u,x)\alpha_{\mathscr{T}_i}(u,y) = \alpha_{\mathscr{T}_i}(h_i,u)^2\alpha_{\mathscr{T}_i}(x,y). \tag{E.12}$$

By assumption on the input to Alg. 2 of the previous iteration, the graph G_i is a multiplicative subgraph of $\mathscr{T}_i$ and hence $G_i(x,y)=\alpha_{\mathscr{T}_i}(x,y)$. Thus, Eqs (E.5) and (E.12) imply

$$G_{i+1}(x,y) = G_i(x,y) - \alpha_{\mathscr{T}_i}(h_i,u)^2\alpha_{\mathscr{T}_i}(x,y) = G_i(x,y) - \alpha_{\mathscr{T}_i}(h_i,u)^2G_i(x,y) = G_i(x,y)(1-\alpha_{\mathscr{T}_i}(h_i,u)^2).$$

Next, we show that $G_{i+1}(x,y)$ is equal to the multiplicative weight $\alpha_{\mathscr{T}_{i+1}}(x,y)$. Let $z_1=x,\dots,z_\kappa=u,\dots,z_K=y$ be the nodes on the path between x and y. By steps 2 and 3 in Algorithm 3, the multiplicative weight $\alpha_{\mathscr{T}_{i+1}}(x,y)$ is equal to

$$\begin{aligned}\alpha_{\mathscr{T}_{i+1}}(x,y) &= \prod_{j=1}^{\kappa-1}\mathscr{T}_{i+1}(z_j,z_{j+1})\prod_{j=\kappa}^{K-1}\mathscr{T}_{i+1}(z_j,z_{j+1})\\ &= \mathscr{T}_i(x,z_2)\sqrt{1-\alpha_{\mathscr{T}_i}(z_2,h_i)^2}\prod_{j=2}^{\kappa-1}\mathscr{T}_i(z_j,z_{z+1})\frac{\sqrt{1-\alpha_{\mathscr{T}_i}(z_{j+1},h_i)^2}}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_j,h_i)^2}}\\ &\times\mathscr{T}_i(y,z_{K-1})\sqrt{1-\alpha_{\mathscr{T}_i}(z_{K-1},h_i)^2}\prod_{j=\kappa}^{K-2}\mathscr{T}_i(z_j,z_{z+1})\frac{\sqrt{1-\alpha_{\mathscr{T}_i}(z_j,h_i)^2}}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_{j+1},h_i)^2}}. \end{aligned} \tag{E.13}$$

Note that

$$\mathscr{T}_i(x,z_2)\sqrt{1-\alpha_{\mathscr{T}_i}(z_2,h_i)^2}\prod_{j=2}^{\kappa-1}\mathscr{T}_i(z_j,z_{z+1})\frac{\sqrt{1-\alpha_{\mathscr{T}_i}(z_{j+1},h_i)^2}}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_j,h_i)^2}}=\sqrt{1-\alpha_{\mathscr{T}_i}(z_\kappa,h_i)^2}\prod_{j=1}^{\kappa-1}\mathscr{T}_i(z_j,z_{z+1})$$

and

$$\mathscr{T}_i(y,z_{K-1})\sqrt{1-\alpha_{\mathscr{T}_i}(z_{K-1},h_i)^2}\prod_{j=\kappa}^{K-2}\mathscr{T}_i(z_j,z_{z+1})\frac{\sqrt{1-\alpha_{\mathscr{T}_i}(z_j,h_i)^2}}{\sqrt{1-\alpha_{\mathscr{T}_i}(z_{j+1},h_i)^2}}$$
$$=\sqrt{1-\alpha_{\mathscr{T}_i}(z_\kappa,h_i)^2}\prod_{j=\kappa}^{K-1}\mathscr{T}_i(z_j,z_{z+1})$$

and thus,

$$\alpha_{\mathscr{T}_{i+1}}(x,y)=\prod_{j=1}^{K-1}\mathscr{T}_i(z_j,z_{z+1})(1-\alpha_{\mathscr{T}_i}(z_\kappa,h_i)^2)=G_{i+1}(x,y).$$

Lastly, we consider case (iii), where $x=h_i$ and $y=v_{i,1}$ or $y=v_{i,2}$. Recall that $v_{i,1},v_{i,2}$ are adjacent to h_i in $\mathscr{T}$ and were removed from the active set. By step 4 of Algorithm 2 and step 1 of Algorithm 3 the edge $G_i(x,y)$ and its corresponding edge $\mathscr{T}_i(x,y)$ have both been updated such that $\mathscr{T}_{i+1}(x,y)=G_{i+1}(x,y)=d\mathscr{T}_i(x,y)$. □

F. Auxiliary Lemmas for Section 5

Proof of Lemma 5.2. We begin by characterizing all the eigenvectors of $L\in\mathbb{R}^{m\times m}$. For any non-terminal node h in the binary symmetric tree $\mathscr{T}$, we denote the set of descendent terminal nodes to the "left" of h by A, the set of descendant terminal nodes to the 'right' of h by B, and the rest of the terminal nodes by C. Let $v_h\in\mathbb{R}^m$ be a vector with

$$(v_h)_i=\begin{cases}1 & i\in A\\ -1 & i\in B\\ 0 & i\in C.\end{cases}$$

We show that for any choice of non-terminal node h, v_h is an eigenvector of L. Since there are $m-1$ non-terminal nodes, this set of eigenvectors, together with the vector of all-ones, forms the full set of all eigenvectors of L.

First, we show that v_h is an eigenvector of the similarity matrix S, and compute the corresponding eigenvalue. For $i\in A$,

$$(Sv_h)_i=\sum_{j\in A}S(i,j)-\sum_{k\in B}S(i,k).$$

Due to the symmetry of the tree $\mathscr{T}$, every terminal node has a similarity of δ^2 to one other terminal node, δ^4 to two other terminal nodes, etc. Thus,

$$\sum_{j\in A} S(i,j) = 1 + \delta^2 + 2\delta^4 + \ldots, + \ldots, |A|\delta^{2\log_2|A|} = \delta^2\left(\frac{1-(2\delta^2)^{\log_2|A|}}{1-2\delta^2}\right) + 1.$$

The similarity between a node $i \in A$ and all nodes $k \in B$ is equal to $\delta^{2(\log|A|+1)}$. Thus,

$$\begin{aligned}\sum_{j\in A} S(i,j) - \sum_{k\in B} S(i,k) &= \delta^2\left(\frac{1-(2\delta^2)^{\log_2|A|}}{1-2\delta^2}\right) + 1 - |A|\delta^{2(\log|A|+1)} \\ &= 1 + \delta^2\left(\frac{1-(2\delta^2)^{\log|A|}(2-2\delta^2)}{1-2\delta^2}\right). \end{aligned} \tag{F.1}$$

The same result with a negative sign holds for $i \in B$. If $i \in C$ then by symmetry $(Sv_h)_i = 0$. Thus v_h is an eigenvector of S with eigenvalue equal to the right side of (F.1). The sum of every row in S is equal to,

$$d_i = \sum_j S(i,j) = 1 + \delta^2 + 2\delta^4 + \ldots + 2^{\log_2 m}\delta^{2\log_2 m} = \delta^2\left(\frac{1-(2\delta^2)^{\log_2 m}}{1-2\delta^2}\right) + 1. \tag{F.2}$$

Let D be the scalar matrix with diagonal elements equal to Eq. (F.2). Combining Eq. (F.2) and Eq. (F.1), we get that v_h is an eigenvector of $L = D - S$ with eigenvalue:

$$\lambda(h) = \delta^2\left(\frac{(2\delta^2)^{\log_2|A|}(2-2\delta^2) - (2\delta^2)^{\log_2 m}}{1-2\delta^2}\right). \tag{F.3}$$

For any Laplacian matrix 0 is an eigenvalue that correspond to the vector of all-ones. Since the eigenvalue $e(h)$ decreases as $|A|$ grows, the two smallest non-zero eigenvalues correspond to $|A| = m/2$ and $|A| = m/4$. The three smallest eigenvalues are thus equal to,

$$\lambda_1 = 0, \qquad \lambda_2 = m^{2\log_2(\delta)+1}, \qquad \lambda_3 = m^{2\log_2(\delta)+1}\left(\frac{1}{2} + \frac{1}{2\delta^2}\right).$$

□

In the following proof, we use similar notations as in the proof of Lemma 5.4.

Proof of Lemma 5.5. For simplicity, let $x = \|S(A_i, B)\|_F^2$ and $y = \|S(A_{i+k}, B)\|_F^2$. To compute the numerator of Eq. (5.14), we set the partial derivative w.r.t. β to 0, which gives

$$\beta^* = \operatorname*{argmin}_\beta \left((1-\beta R_i)^2 x + (1-\beta R_{i+k})^2 y\right) = \frac{R_i x + R_{i+k} y}{R_i^2 x + R_{i+k}^2 y}.$$

Plugging β^* back into the numerator of Eq. (5.14) gives

$$\min_{\beta}\Big((1-\beta R_i)^2 x + (1-\beta R_{i+k})^2 y\Big) = \frac{xy(R_i - R_{i+k})^2}{R_i^2 x + R_{i+k}^2 y}.$$

Observe that $R_{i+k} = R_i S(h_i, h_{i+k})^2$. Thus, the above expression further simplifies to

$$\frac{xy(R_i - R_{i+k})^2}{R_i^2 x + R_{i+k}^2 y} = \frac{xyR_i^2(1-S(h_i,h_{i+k})^2)^2}{R_i^2(x + S(h_i,h_{i+k})^4 y)} = \frac{xy(1-S(h_i,h_{i+k})^2)^2}{x + S(h_i,h_{i+k})^4 y}.$$

Since $\|S(A_i, B)\|_F^2 + \|S(A_{i+k}, B)\|_F^2 = x + y$, the LHS of (5.14) is equal to

$$\frac{xy(1-S(h_i,h_{i+k})^2)^2}{(x+y)(x+S(h_i,h_{i+k})^4 y)}. \tag{F.4}$$

Recall from Eqs (2.2) and (3.1) that for any $1 \le i \le N-1$, $S(h_i, h_{i+k}) < \xi < 1$. It follows that

$$\frac{xy(1-S(h_i,h_{i+k})^2)^2}{(x+y)(x+S(h_i,h_{i+k})^4 y)} \ge \frac{xy(1-\xi^2)^2}{(x+y)^2} \ge \frac{xy(1-\xi^2)^2}{(2\max(x,y))^2} = \frac{(1-\xi^2)^2 \min(x,y)}{4\max(x,y)}. \tag{F.5}$$

Next, we simplify the term $\frac{\min(x,y)}{\max(x,y)}$ in Eq. (F.5). Note that h_{i+k} separates A_i and A_{i+k} from B, see ilustration in Figure 5. Thus, we can rewrite $\min(x, y)$ as

$$\begin{aligned}
\min(x,y) &= \min(\|S(A_i,B)\|_F^2, \|S(A_{i+k},B)\|_F^2) \\
&= \min(\|S(A_i,h_{i+k})S(h_{i+k},B)\|_F^2, \|S(A_{i+k},h_{i+k})S(h_{i+k},B)\|_F^2) \\
&= \min(\|S(A_i,h_{i+k})\|^2\|S(h_{i+k},B)\|^2, \|S(A_{i+k},h_{i+k})\|^2\|S(h_{i+k},B)\|_F^2) \\
&= \min(\|S(A_i,h_{i+k})\|^2, \|S(A_{i+k},h_{i+k})\|^2)\cdot\|S(h_{i+k},B)\|^2.
\end{aligned}$$

Similarly, $\max(x, y) = \max(\|S(A_i, h_{i+k})\|^2, \|S(A_{i+k}, h_{i+k})\|^2) \cdot \|S(h_{i+k}, B)\|^2$. Thus,

$$\frac{\min(x,y)}{\max(x,y)} = \frac{\min(\|S(A_i,h_{i+k})\|^2, \|S(A_{i+k},h_{i+k})\|^2)}{\max(\|S(A_i,h_{i+k})\|^2, \|S(A_{k+1},h_{i+k})\|^2)}.$$

Next, we provide lower and upper bounds on the terms $\|S(A_i, h_{i+k})\|^2$ and $\|S(A_{i+1}, h_{i+k})\|^2$. By Eq. (2.2), the similarity between the nodes in A_i, A_{i+k} and h_{i+k} is bounded by ξ. It follows that

$$\max(\|S(A_i,h_{i+k})\|^2, \|S(A_{i+k},h_{i+k})\|^2) \le \max(|A_i|,|A_{i+k}|)\xi^2 \le m\xi^2. \tag{F.6}$$

For a lower bound, we apply [26, Lemma 4.5]. Given the terminal nodes of a clan A, and the root of a clan h, the lemma bounds the norm of $S(A,h)$ by,

$$\|S(A,h)\|_F^2 \geq \begin{cases} (2\delta^2)^{\log|A|} & \delta^2 \leq 0.5 \\ 2\delta^2 & \delta^2 > 0.5 \end{cases} \geq \begin{cases} (2\delta^2)^{\log m} & \delta^2 \leq 0.5 \\ 2\delta^2 & \delta^2 > 0.5. \end{cases}$$

There are $k+1$ edges between the root of A_i and h_{i+k}, and one edge between the root of A_{i+k} and h_{i+k}. Thus,

$$\min(\|S(A_i,h_{i+k})\|^2, \|S(A_{i+k},h_{i+k})\|^2) \geq \begin{cases} (2\delta^2)^{\log m}\delta^{2(k+1)} & \delta^2 \leq 0.5 \\ 2\delta^{(2k+2)} & \delta^2 > 0.5. \end{cases} \tag{F.7}$$

Plugging Eqs (F.6), (F.7) into (F.5) concludes the proof. □

Proof of Lemma 5.6. The lemma is a small variation over the known lower bound for ratio of sums, $\frac{\sum_i a_i}{\sum_i b_i} \geq \min_i \frac{a_i}{b_i}$. For an even number of elements, we can merge non overlapping pairs of consecutive elements such that $\tilde{a}_i = a_{2i} + a_{2i+1}$ and $\tilde{b}_i = b_{2i} + b_{2i+1}$. Applying the standard bound for ratio of sums for $\tilde{a}_i$ and $\tilde{b}_i$ gives,

$$\frac{\sum_i \tilde{a}_i}{\sum_i \tilde{b}_i} \geq \min_i \frac{\tilde{a}_i}{\tilde{b}_i} = \min_i \frac{a_{2i}+a_{2i+1}}{b_{2i}+b_{2i+1}} \geq \min_{i\neq j;|i-j|\leq 2} \frac{a_i+a_j}{b_i+b_j}.$$

For an odd number of elements, we can merge the first three elements $i = 0,1,2$. The rest will be merged into consecutive pairs.

$$\frac{\sum_i a_i}{\sum_i b_i} \geq \min\left\{ \frac{a_0+a_1+a_2}{b_0+b_1+b_2}, \frac{\sum_{i\geq 2}(a_{2i}+a_{2i+1})}{\sum_{i\geq 2}(b_{2i}+b_{2i+1})} \right\}$$

The ratio for elements $i = 0,1,2$ can be bounded by the minimum ratio over all pairs $i,j \in \{0,1,2\}$. Thus,

$$\frac{\sum_i a_i}{\sum_i b_i} \geq \min\left\{ \min_{i\neq j\in\{0,1,2\}} \frac{a_i+a_j}{b_i+b_j}, \frac{\sum_{i\geq 2}(a_{2i}+a_{2i+1})}{\sum_{i\geq 2}(b_{2i}+b_{2i+1})} \right\} \geq \min_{i\neq j;|i-j|\leq 2} \frac{a_i+a_j}{b_i+b_j}$$

□

LEMMA F.1. Let $X, X' \in \mathbb{R}^{m\times n}$ and let $y, y' > 0$. Assume that $\|X'\|_F \leq y'$, then

$$\left\| \frac{X}{y} - \frac{X'}{y'} \right\|_F \leq \frac{1}{y}(\|X-\hat{X}\|_F + |y-\hat{y}|). \tag{F.8}$$

Proof. By definition,

$$\left\| \frac{X}{y} - \frac{\hat{X}}{\hat{y}} \right\|_F = \left\| \frac{Xy' - X'y}{yy'} \right\|_F = \left\| \frac{y'(X-X')}{yy'} + \frac{X'(y'-y)}{yy'} \right\|_F$$

By the triangle inequality

$$\left\|\frac{X}{y}-\frac{\hat{X}}{\hat{y}}\right\|_F \leq \frac{1}{y}\|X-X'\|_F + \frac{|y'-y|}{y}\cdot\frac{\|X'\|_F}{y'}$$

Since $\|X'\|_F \leq y'$ the lemma follows. □

LEMMA F.2. Let X and Y be two matrices and let $\hat{X}$ and $\hat{Y}$ be their corresponding noisy estimates. Then,

$$\left|\|X-Y\|_F - \|\hat{X}-\hat{Y}\|_F\right| \leq \|X-\hat{X}\|_F + \|Y-\hat{Y}\|_F.$$

Proof. Assume that $\|X-Y\|_F \geq \|\hat{X}-\hat{Y}\|_F$. In this case

$$\left|\|X-Y\|_F - \|\hat{X}-\hat{Y}\|_F\right| = \|X-Y\|_F - \|\hat{X}-\hat{Y}\|_F \leq \|X-Y-\hat{X}+\hat{Y}\|_F \leq \|X-\hat{X}\|_F + \|Y-\hat{Y}\|_F.$$

Alternatively, if $\|X-Y\|_F \leq \|\hat{X}-\hat{Y}\|_F$ we have

$$\left|\|X-Y\|_F - \|\hat{X}-\hat{Y}\|_F\right| = \|\hat{X}-\hat{Y}\|_F - \|X-Y\|_F \leq \|\hat{X}-\hat{Y}-X+Y\|_F \leq \|\hat{X}-X\|_F + \|\hat{Y}-Y\|_F.$$

□

LEMMA F.3. Let $X \in \mathbb{R}^{n_1\times n_2}, Y \in \mathbb{R}^{n_2\times n_3}, Z \in \mathbb{R}^{n_3\times n_4}$ be three matrices and let $\hat{X},\hat{Y},\hat{Z}$ be there corresponding estimates. Then

$$\|XYZ-\hat{X}\hat{Y}\hat{Z}\|_F \leq \|X\|_F\|Y\|_F\|Z-\hat{Z}\|_F + \|\hat{Z}\|_F\|Y\|_F\|X-\hat{X}\|_F + \|\hat{Z}\|_F\|\hat{X}\|_F\|Y-\hat{Y}\|_F$$

Proof.

$$\|XYZ-\hat{X}\hat{Y}\hat{Z}\|_F = \|XYZ-XY\hat{Z}+XY\hat{Z}-\hat{X}\hat{Y}\hat{Z}\|_F \leq \|X\|_F\|Y\|_F\|Z-\hat{Z}\|_F + \|\hat{Z}\|_F\|XY-\hat{X}\hat{Y}\|_F \tag{F.9}$$

Focusing on $\|XY-\hat{X}\hat{Y}\|_F$ we have that

$$\|XY-\hat{X}\hat{Y}\|_F = \|XY-\hat{X}Y+\hat{X}Y-\hat{X}\hat{Y}\|_F \leq \|X-\hat{X}\|_F\|Y\|_F + \|\hat{X}\|\|Y-\hat{Y}\|_F$$

Combining the two bounds gives,

$$\|XYZ-\hat{X}\hat{Y}\hat{Z}\|_F \leq \|X\|_F\|Y\|_F\|Z-\hat{Z}\|_F + \|\hat{Z}\|_F\|Y\|_F\|X-\hat{X}\|_F + \|\hat{Z}\|_F\|\hat{X}\|_F\|Y-\hat{Y}\|_F$$

□

LEMMA F.4. Let S denote a rank one matrix and $\hat{S}$ its noisy estimate. We denote by $u,\hat{u}$ their respective leading left singular vectors. If $\|S-\hat{S}\|_F \leq 0.5\|S\|_F$ then

$$\|uu^T-\hat{u}\hat{u}^T\|_F^2 \leq \frac{50\|S-\hat{S}\|_F^2}{\|S\|_F^2}.$$

Proof.

$$\|uu^T - \hat{u}\hat{u}^T\|_F^2 = \sum_{ij}(uu^T - \hat{u}\hat{u}^T)_{ij}^2 = \sum_{ij}(uu^T)_{ij}^2 + \sum_{ij}(\hat{u}\hat{u}^T)_{ij}^2 - 2\sum_{ij}(uu^T)_{ij}(\hat{u}\hat{u}^T)_{ij}$$
$$= \|u\|^4 + \|\hat{u}\|^4 - 2\sum_i u_i\hat{u}_i \sum_j u_j\hat{u}_j = 2(1 - (u^T\hat{u})^2) = 2\sin^2(u,\hat{u}). \tag{F.10}$$

We apply a variant of the Davis-Kahan theorem for non square matrices [62, Theorem 3). The perturbation of the leading singular vector is bounded by

$$\sin(u,\hat{u}) \le \frac{2(2\sigma_1(S) + \|S - \hat{S}\|)\|S - \hat{S}\|}{\sigma_1^2(S) - \sigma_2^2(S)},$$

where $\sigma_1(S)$ and $\sigma_2(S)$ are the two leading singular values of S. Since S is rank one, $\sigma_1(S) = \|S\| = \|S\|_F$ and $\sigma_2(S) = 0$. In addition, we assumed that $\|S - \hat{S}\|_F \le 0.5\|S\|_F$ and hence

$$\sin(u,\hat{u}) \le \frac{5\|S\|_F\|S - \hat{S}\|_F}{\|S\|_F^2} = \frac{5\|S - \hat{S}\|_F}{\|S\|_F}. \tag{F.11}$$

Combining Eqs (F.10), (F.11) concludes the proof. □

Proof of Lemma 5.8. Let $d(e)$ denote the score of the edge e computed by the exact similarity matrix S as defined in (3.6). We denote by $\hat{d}(e)$ the score computed by the noisy estimate of the similarity $\hat{S}$. The difference between $d(e)$ and $\hat{d}(e)$ is equal to

$$|d(e) - \hat{d}(e)| = \left| \frac{\|S(A,B) - \alpha^* u_A u_B^T\|_F}{\|S(A,B)\|_F} - \frac{\|\hat{S}(A,B) - \beta^*\hat{u}_A\hat{u}_B^T\|_F}{\|\hat{S}(A,B)\|_F} \right|, \tag{F.12}$$

where,

$$\alpha^* = \operatorname*{argmin}_\alpha \|S(A,B) - \alpha u_A u_B^T\|_F \qquad \beta^* = \operatorname*{argmin}_\beta \|\hat{S}(A,B) - \beta\hat{u}_A\hat{u}_B^T\|_F.$$

We apply Lemma F.1 with

$$X = \|S(A,B) - \alpha^* u_A u_B^T\|_F, \quad y = \|S(A,B)\|_F, \quad \hat{X} = \|\hat{S}(A,B) - \beta^*\hat{u}_A\hat{u}_B^T\|_F, \quad \hat{y} = \|\hat{S}(A,B)\|_F,$$

where we note that here X and $\hat{X}$ are scalars. Lemma F.1 requires that $0 < |\hat{X}| \le \hat{y}$, which holds trivially. Applying Lemma F.1 to (F.12) yields,

$$|d(e) - \hat{d}(e)| \le \frac{1}{\|S(A,B)\|_F}\Big(\Big|\|S(A,B) - \alpha^* u_A u_B^T\|_F - \|\hat{S}(A,B) - \beta^*\hat{u}_A\hat{u}_B^T\|_F\Big|$$
$$+\Big|\|S(A,B)\|_F - \|\hat{S}(A,B)\|_F\Big|\Big). \tag{F.13}$$

Next, setting $X = S(A,B), \hat{X} = \hat{S}(A,B), Y = \alpha^* u_A u_B^T$ and $\hat{Y} = \beta^* \hat{u}_A \hat{u}_B^T$, by Lemma F.2,

$$\begin{aligned}|d(e) - \hat{d}(e)| &\le \frac{1}{\|S(A,B)\|_F} \Big(\|S(A,B) - \hat{S}(A,B)\|_F + \|\alpha^* u_A u_B^T - \beta^* \hat{u}_A \hat{u}_B^T\|_F \\ &\qquad + \big| \|S(A,B)\|_F - \|\hat{S}(A,B)\|_F \big| \Big) \\ &\le \frac{1}{\mathscr{D}} \Big(2\|S(A,B) - \hat{S}(A,B)\|_F + \|\alpha^* u_A u_B^T - \beta^* \hat{u}_A \hat{u}_B^T\|_F \Big). \end{aligned} \tag{F.14}$$

where the second inequality is due to the reverse triangle inequality and the definition of $\mathscr{D}$.

We focus on the term $\|\alpha^* u_A u_B^T - \beta^* \hat{u}_A \hat{u}_B^T\|_F$ in Eq. (F.14). The values of α^*, β^* are obtained via least square between the elements of $S(A,B), \hat{S}(A,B)$ and $u_A u_B^T, \hat{u}_A \hat{u}_B^T$, respectively. For α^*, the least squares solution is

$$\alpha^* = \frac{1}{\|S(A,B)\|_F^2} \sum_{i,j} S(A,B)_{ij} (u_A u_B^T)_{ij} = \frac{1}{\|S(A,B)\|_F^2} u_A^T S(A,B) u_B, \tag{F.15}$$

where a similar expression holds for β^*. Multiplying α^* and β^* by $u_A u_B^T$ and $\hat{u}_A \hat{u}_B^T$ gives,

$$\alpha^* u_A u_B - \beta^* \hat{u}_A \hat{u}_B = \frac{1}{\|S(A,B)\|_F^2} u_A u_A^T S(A,B) u_B u_B^T - \frac{1}{\|\hat{S}(A,B)\|_F^2} \hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T. \tag{F.16}$$

Next, we apply Lemma F.1 with $X = u_A u_A^T S(A,B) u_B u_B^T$, $y = \|S(A,B)\|_F^2$, $\hat{X} = \hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T$ and $\hat{y} = \|\hat{S}(A,B)\|_F^2$. The condition for Lemma F.1 is that $\|\hat{X}\|_F \le \hat{y}$, which holds since

$$\|\hat{X}\|_F = \|\hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T\|_F \le \|\hat{u}_A \hat{u}_A^T\|_F \|\hat{S}(A,B)\|_F \|\hat{u}_B \hat{u}_B^T\|_F \le \|\hat{S}(A,B)\|_F = \hat{y}.$$

Applying Lemma F.1 to (F.16) gives

$$\begin{aligned}\|\alpha^* u_A u_B - \beta^* \hat{u}_A \hat{u}_B\|_F \le \frac{1}{\|S(A,B)\|_F^2} \Big(&\|u_A u_A^T S(A,B) u_B u_B^T - \hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T\|_F \cdot \\ &+ \big| \|S(A,B)\|_F^2 - \|\hat{S}(A,B)\|_F^2 \big| \Big). \end{aligned} \tag{F.17}$$

Denote

$$\begin{aligned} \varepsilon(A,B) &= \hat{S}(A,B) - S(A,B) & \varepsilon(C_1,C_2) &= \hat{S}(C_1,C_2) - S(C_1,C_2) \\ \varepsilon_A &= \hat{u}_A \hat{u}_A^T - u_A u_A^T & \varepsilon_B &= \hat{u}_B \hat{u}_B^T - u_B u_B^T. \end{aligned}$$

Equipped with the above notations, we bound the first term in the numerator of Eq. (F.17) using Lemma F.3 where $X = u_A u_A^T$, $Y = S(A,B)$, and $Z = u_B u_B^T$,

$$\begin{aligned}&\|u_A u_A^T S(A,B) u_B u_B^T - \hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T\|_F \\ &\quad \le \|u_A u_A^T\|_F \|S(A,B)\|_F \|\varepsilon_B\|_F + \|\hat{u}_B \hat{u}_B^T\|_F \|S(A,B)\|_F \|\varepsilon_A\|_F + \|\hat{u}_B \hat{u}_B^T\|_F \|\hat{u}_A \hat{u}_A^T\|_F \|\varepsilon(A,B)\|_F \end{aligned}$$

Since $\|u_A u_A^T\|_F, \|\hat{u}_B \hat{u}_B^T\|_F \leq 1$ we get,

$$\|u_A u_A^T S(A,B) u_B u_B^T - \hat{u}_A \hat{u}_A^T \hat{S}(A,B) \hat{u}_B \hat{u}_B^T\|_F \leq \|S(A,B)\|_F (\|\varepsilon_A\|_F + \|\varepsilon_B\|_F) + \|\varepsilon(A,B)\|_F. \tag{F.18}$$

The matrices $\varepsilon_A, \varepsilon_B$ are submatrices of $\hat{u}\hat{u}^T - uu^T$ and hence $\|\varepsilon_A\|_F, \|\varepsilon_B\|_F \leq \|\hat{u}\hat{u}^T - uu^T\|_F$. Applying Lemma F.4 gives

$$\|\varepsilon_A\|_F + \|\varepsilon_B\|_F \leq 2\|uu^T - \hat{u}\hat{u}^T\|_F \leq \frac{10\sqrt{2}\|\varepsilon(C_1,C_2)\|_F}{\|S(C_1,C_2)\|_F} \leq \frac{10\sqrt{2}\|\varepsilon(C_1,C_2)\|_F}{\mathscr{D}}. \tag{F.19}$$

Combining Eqs (F.14), (F.17),(F.18) and (F.19) yields

$$|d(e) - \hat{d}(e)| \leq \frac{1}{\mathscr{D}} \Bigg(2\|\varepsilon(A,B)\|_F + \frac{1}{\|S(A,B)\|_F^2} \Bigg(\big| \|S(A,B)\|_F^2 - \|\hat{S}(A,B)\|_F^2 \big| + \|\varepsilon(A,B)\|_F + \frac{10\sqrt{2}\|S(A,B)\|_F \|\varepsilon(C_1,C_2)\|_F}{\mathscr{D}} \Bigg) \Bigg). \tag{F.20}$$

We have that

$$\begin{aligned} \big| \|S(A,B)\|_F^2 - \|\hat{S}(A,B)\|_F^2 \big| &= \big| \|S(A,B)\|_F - \|\hat{S}(A,B)\|_F \big| (\|S(A,B)\|_F + \|\hat{S}(A,B)\|_F) \\ &\leq 2.5\|\varepsilon(A,B)\|_F \|S(A,B)\|_F, \end{aligned} \tag{F.21}$$

where the inequality is due to the reverse triangle inequality and our assumption $\|\varepsilon(A,B)\|_F \leq 0.5\|S(A,B)\|_F$ which implies $\|\hat{S}(A,B)\|_F \leq 1.5\|S(A,B)\|_F$. Combining (20) and (F.21), we get

$$\begin{aligned} |d(e) - \hat{d}(e)| &\leq \frac{1}{\mathscr{D}} \Bigg(2\|\varepsilon(A,B)\|_F + \frac{1}{\|S(A,B)\|_F^2} \times \\ &\qquad \Bigg(\|\varepsilon(A,B)\|_F \left(2.5\|S(A,B)\|_F + 1\right) + \frac{10\sqrt{2}\|\varepsilon(C_1,C_2)\|_F \|S(A,B)\|_F}{\mathscr{D}} \Bigg) \Bigg) \\ &\leq \|\varepsilon(A,B)\|_F \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1}{\mathscr{D}^3} \right) + \|\varepsilon(C_1,C_2)\|_F \frac{10\sqrt{2}}{\mathscr{D}^3} \\ &\leq \|S - \hat{S}\|_F \left(\frac{2}{\mathscr{D}} + \frac{2.5}{\mathscr{D}^2} + \frac{1 + 10\sqrt{2}}{\mathscr{D}^3} \right), \end{aligned}$$

which concludes the proof. □

G. The value of $\mathscr{D}$

In our analysis, we provide in Theorem 5.3 a finite sample guarantee for the merging step. Given two trees $\mathscr{T}_1, \mathscr{T}_2$ with terminal nodes C_1, C_2, our goal is to find the correct merging edge $e^*(A,B)$ in $\mathscr{T}_1$

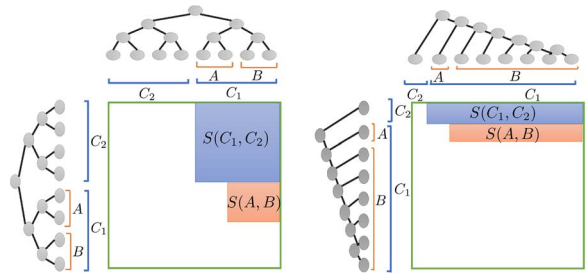


FIG. G4. Illustration of $\mathscr{D}$ for balanced binary and caterpillar trees.

where $A \cup B = C_1$. The obtained guarantee depends on the value of $\mathscr{D}$, given by

$$\mathscr{D} = \min\{\|S(A,B)\|_F, \|S(C_1,C_2)\|_F\}.$$

For trees that are balanced, this value is of order of m^2. For illustration, consider the case of a balanced binary tree, illustrated on the left panel of Figure G4. For this case, the number of elements of $S(A,B)$ is $m^2/16$. Assuming that all edges have similarity δ, the value of each element is $\delta^{2\log m}$. On the right panel, we illustrate the case of a caterpillar tree, here the number of elements in both $S(C_1,C_2)$ and $S(A,B)$ is linear in m.

H. Proof of Lemma 5.3

Theorem 5.2 guarantees that for n large enough, the partitioning step yields $\mathscr{T}_1$ and $\mathscr{T}_2$ that both satisfy the molecular clock model. By the definition of η, the number of terminal nodes $|C_1|, |C_2|$ in both trees is at least $m/(1+\eta)$.

Consider the matrix $S(C_1,C_2)$ that contains the similarity between the terminal nodes of $\mathscr{T}_1, \mathscr{T}_2$. Under the assumption of the molecular clock, the matrix $S(C_1,C_2)$ is constant, with values $\exp(-r(\mathscr{T}))$. Thus,

$$\|S(C_1,C_2)\|_F = \exp(-r(\mathscr{T}))\sqrt{|C_1||C_2|}.$$

Assume that the ratio $|C_1|/|C_2| = \eta_{12}$. In that case

$$|C_1||C_2| = \frac{\eta_{12} m^2}{(1+\eta_{12})^2}.$$

Since the right-hand side is monotonically decreasing for $\eta_{12} > 1$, and $\eta_{12} \le \eta$. Then,

$$|C_1||C_2| \ge \frac{\eta m^2}{(1+\eta)^2}.$$

and thus,

$$\|S(C_1,C_2)\|_F \ge \exp(-r(\mathscr{T}))\frac{\sqrt{\eta} m}{1+\eta}.$$

For $S(A,B)$, we replace m with the lower bound on the size $|A \cup B| = |C_1| \geq m/(1+\eta)$. The diameter of the new tree whose terminal nodes are $A \cup B$ is at most $r(t) - 2\log(\delta)$ and hence,

$$\|S(A,B)\|_F \geq \frac{\exp(-r(\mathscr{T}))}{\delta^2}\sqrt{\eta}\max\left\{1, \frac{m}{(1+\eta)^2}\right\}.$$

The two bounds for $\|S(A,B)\|_F$ and $\|S(C_1,C_2)\|_F$ yield

$$\mathscr{D} \geq \exp(-r(\mathscr{T}))\sqrt{\eta}\max\left\{1, \frac{m}{(1+\eta)^2}\right\}.$$

I. Proof of Theorem 5.4

The guarantees for a single partition and merging step presented in Theorems 5.2 and 5.3 are based on the respective two lemmas 5.7 and 5.1 that provide a guarantee for STDR under the condition on the accuracy $\|S - \hat{S}\|$. Let C_1, C_2 denote the first partition. For any additional partition and merging iterations (say $(C_{11}, C_{12}), (C_{21}, C_{22})$ etc.), similar guarantees hold for the submatrix $\|S(C_1,C_2) - \hat{S}(C_1,C_2)\|$. Note, that for any submatrix $S(C_1,C_2)$ of S,

$$\|S(C_1,C_2) - \hat{S}(C_1,C_2)\| \leq \|S - \hat{S}\|.$$

However, the expressions in the two bounds in Lemmas 5.7 and 5.1 depend on parameters such as $r(\mathscr{T}), h(\mathscr{T})$ (for the partitioning step) and $\mathscr{D}$ (for the merging step), which might be different for the subtrees. Thus, our goal is to further develop the bounds in the two lemmas such that they hold for any subtree that is partitioned during the STDR algorithm. We do so separately for the partitioning and merging steps.

PARTITIONING STEP. Lemma 5.1 guarantees that the partitioning step is accurate if

$$\|S - \hat{S}\| \leq \frac{\sqrt{m}e^{-r(\mathscr{T})}}{\sqrt{\eta}2^{3/2}(\sqrt{m}+1)}\min\left\{1, \frac{1}{1+\eta}\left(e^{r(\mathscr{T})-h(\mathscr{T})} - 1\right)\right\}.$$

For the molecular lock model, $h(\mathscr{T})$ is the distance between the terminal nodes and the root, and thus $r(\mathscr{T}) = 2h(\mathscr{T})$. The expression $\exp(h(\mathscr{T}))$ is the inverse of the similarity between the terminal nodes and the root which is bounded by $1/\xi$. Thus,

$$e^{r(\mathscr{T})-e(h(\mathscr{T}))} - 1 = e^{h(\mathscr{T})} - 1 \geq \exp(-\log\xi) - 1 = \frac{1-\xi}{\xi}.$$

In addition, $e^{-r(\mathscr{T})} \leq e^{-r(\mathscr{T}(C_1))}$ for any subtree $\mathscr{T}(C_1)$ of $\mathscr{T}$. Thus, if $\|S - \hat{S}\|$ satisfies

$$\|S - \hat{S}\| \leq \frac{\sqrt{m}e^{-r(\mathscr{T})}}{\sqrt{\eta}2^{3/2}(\sqrt{m}+1)}\frac{1-\xi}{\xi(1+\eta)},$$

then the partitioning step is guaranteed to yield an accurate partition for all STDR iterations.

MERGING STEP. The allowed error in the estimate of S is equal to

$$\|S-\hat{S}\|_F \leq \frac{1}{2}\left(\frac{2}{\mathscr{D}}+\frac{2.5}{\mathscr{D}^2}+\frac{1+10\sqrt{2}}{\mathscr{D}^3}\right)^{-1}\frac{\delta^3(1-\xi^2)}{\sqrt{2m}\xi^2},$$

The only parameter that decreases as we move from the full tree to a subtree is the value of $\mathscr{D}$. We use Lemma 5.3, which bounds the value of $\mathscr{D}$, for a tree of size m via,

$$\mathscr{D} \geq \exp(-r(\mathscr{T}))\max\left\{1,\frac{\sqrt{\eta}m}{(1+\eta)^2}\right\}.$$

Here, we replace m with τ—the user provided parameter for the minimal tree size to be partitioned. Thus, for guaranteed merging correctness for all partitions we have,

$$\|S-\hat{S}\|_F \leq \frac{1}{2}\left(\frac{2}{\mathscr{D}(\tau)}+\frac{2.5}{\mathscr{D}(\tau)^2}+\frac{1+10\sqrt{2}}{\mathscr{D}(\tau)^3}\right)^{-1}\frac{\delta^3(1-\xi^2)}{\sqrt{2m}\xi^2},$$

where

$$\mathscr{D}(\tau) = \exp(-r(\mathscr{T}))\max\left\{1,\frac{\sqrt{\eta}\tau}{(1+\eta)^2}\right\}.$$

Combining these results with the probabilistic bound on the estimate $\hat{S}$ in Eq. (5.12) (partitioning) and (5.22) (merging) conclude the proof.

J. Proof of merging complexity

Proof of Lemma 6.1. The complexity of merging two subtrees with k terminal nodes each is composed of two parts:

1. Compute the leading singular vector of the matrix $S(C_1,C_2) \in \mathbb{R}^{k\times k}$, which takes $\mathscr{O}(k^2)$ operations.
2. Compute the score for every edge as in Eq. (3.6). The number of operations required for the least-squares operation in the numerator of Eq. (3.6), as well as computing the Frobenius norms in the numerator and denominator is proportional to the number of elements in $S(A(e),B(e))$. Thus, the total complexity of computing the score for all edges in $\mathscr{T}_1$ (and similarly $\mathscr{T}_2$) is $O(\sum_{e\in\mathscr{T}_1}|A(e)||B(e)|)$.
3. In Lemma J.1 we prove that for any tree $\mathscr{T}_1$ with k terminal nodes that follows the CBM model with η bounded from below by η_0, $\sum_{e\in\mathscr{T}}|A(e)||B(e)| = \mathscr{O}(k^2\log k)$.

Thus, the total computational complexity of the merging is $\mathscr{O}(k^2\log k)$ as well, which concludes the proof. □

LEMMA J.1. Let T be the set of trees with k terminal nodes that follows the CBM model with η bounded from below by η_0. Denote $C(k) = \max_{\mathscr{T}\in T}\sum_{e\in\mathscr{T}}|A(e)||B(e)|$. Then, $C(k) = \mathscr{O}(k^2\log k)$.

Proof. For a tree $\mathscr{T}$, we denote $\tilde{C}(\mathscr{T}) = \sum_{e\in\mathscr{T}} |A(e)||B(e)|$, and note that $C(k) = \max_{\mathscr{T}\in T} \tilde{C}(\mathscr{T})$. We also denote $\tilde{\ell}(\mathscr{T})$ is the sum for every edge we add the number of leaves that are not on the side of the root, explicitly,

$$\tilde{\ell}(\mathscr{T}) = \sum_{e\in\mathscr{T}} |A(e)|.$$

We have the following recurrence formula for $\ell(\mathscr{T})$ given that $\mathscr{T}$ is split to $\mathscr{T}_1$ and $\mathscr{T}_2$:

$$\tilde{\ell}(\mathscr{T}) = \tilde{\ell}(\mathscr{T}_1) + \tilde{\ell}(\mathscr{T}_2) + |\mathscr{T}_1| + |\mathscr{T}_2| = \tilde{\ell}(\mathscr{T}_1) + \tilde{\ell}(\mathscr{T}_2) + |\mathscr{T}|,$$

where $|\mathscr{T}|$ denotes the number of terminal nodes in $\mathscr{T}$.

Given $\ell(\mathscr{T}_1)$ and $\ell(\mathscr{T}_2)$, $C(\mathscr{T})$ can be written as

$$\begin{aligned}\tilde{C}(\mathscr{T}) &= 2|\mathscr{T}_1||\mathscr{T}_2| + \sum_{e\in\mathscr{T}_1} |A_{\mathscr{T}_1}(e)|(|B_{\mathscr{T}_1}(e) + |\mathscr{T}_2|) + \sum_{e\in\mathscr{T}_2} |A_{\mathscr{T}_2}(e)|(|B_{\mathscr{T}_2}(e) + |\mathscr{T}_1|)\\ &= 2|\mathscr{T}_1||\mathscr{T}_2| + \tilde{C}(\mathscr{T}_1) + \tilde{\ell}(\mathscr{T}_1)|\mathscr{T}_2| + \tilde{C}(\mathscr{T}_2) + \tilde{\ell}(\mathscr{T}_2)|\mathscr{T}_1|\end{aligned}$$

Next, we define $\ell(k) = \max_{\mathscr{T}\in T} \tilde{\ell}(\mathscr{T})$. From Lemma J.3 we have that $\ell(k) \le c_\ell k \log k$ for some $c_\ell \in \mathbb{R}$.

Now, we prove that $C(k) < c_c k^2 \log k$. From the definition of $C(k)$, we have that

$$\begin{aligned}C(k_0) &= \max_{\mathscr{T}\in T} \tilde{C}(\mathscr{T}) = \max_{\mathscr{T}_1,T_2} \left(2|\mathscr{T}_1||\mathscr{T}_2| + \tilde{C}(\mathscr{T}_1) + \tilde{\ell}(\mathscr{T}_1)|\mathscr{T}_2| + \tilde{C}(\mathscr{T}_2) + \tilde{\ell}(\mathscr{T}_2)|\mathscr{T}_1|\right)\\ &\le \max_{\alpha_0\le\alpha<0.5} \left(2\alpha(1-\alpha)k_0^2 + C(\alpha k_0) + \ell(\alpha k_0)(1-\alpha)k_0 + C((1-\alpha)k_0) + \ell((1-\alpha)k_0)\alpha k_0\right)\end{aligned}$$

From Lemma J.3, we have,

$$C(k_0) \le \max_{\alpha_0\le\alpha<0.5} \left(C(\alpha k_0) + C((1-\alpha)k_0) + ck_0^2 \log k_0\right),$$

for some $c \in \mathbb{R}$. Using Lemma J.2 we conclude the proof. □

LEMMA J.2. Let $C(k)$ be a function that satisfies the recurrence equation

$$C(k) \le \max_{\alpha_0\le\alpha<0.5} C(\alpha k) + C((1-\alpha)k) + ck^2 \log k$$

for some $0 < \alpha_0 \le 0.5$ independent of k. Then $C(k) = \mathcal{O}(k^2 \log k)$.

Proof. We prove this by induction. The base of the induction is trivial since for small k, there is always a constant C_c such that $C(k) \leq C_c k^2 \log k$. Assume for all $k < k_0$ we have that $C(k) \leq C_c k^2 \log k$. Then,

$$\begin{aligned} C(k_0) &\leq \max_{\alpha_0 \leq \alpha < 0.5} C(\alpha k) + C((1-\alpha)k) + ck^2 \log k \\ &\leq \max_{\alpha_0 \leq \alpha < 0.5} C_c \alpha^2 k_0^2 \log(\alpha k_0) + C_c (1-\alpha)^2 k_0^2 \log((1-\alpha)k_0) + ck_0^2 \log k_0 \\ &\leq k_0^2 \max_{\alpha_0 \leq \alpha < 0.5} (C_c \alpha^2 + C_c (1-\alpha)^2 + c) \log k_0 + C_c \alpha^2 \log(\alpha) + C_c (1-\alpha)^2 \log(1-\alpha). \end{aligned}$$

For large enough C_c we have that $C(k_0) \leq C_c k_0^2 \log k_0$, which concludes the proof. □

LEMMA J.3. $\ell(k) = \mathcal{O}(k \log k)$, or, explicitly, there is $c_\ell \in \mathbb{R}$ such that

$$\ell(k) = c_\ell k \log k$$

REMARK J.1. In case the tree is split at each level with ratio exactly η, the value of $\ell(k)$ can be derived directly from the Akra-Bazzi method [2].

Proof. We prove this by induction. The base of the induction, the proof is trivial. Assume $\tilde{\ell}(k)) \leq c_\ell k \log k$ for all $k < k_0$ we show that $\tilde{\ell}(k_0) \leq c_\ell k_0 \log k_0$.

$$\tilde{\ell}(k_0) = \max \ell(T) = \max_{\mathcal{T}_1, T_2} \left(\ell(\mathcal{T}_1) + \ell(T_2) + k_0\right) \leq \max_{\alpha_0 \leq \alpha < 0.5} \left(\tilde{\ell}(\alpha k_0) + \tilde{\ell}((1-\alpha)k_0)\right) + k_0$$

using the induction assumption, we have

$$\tilde{\ell}(k_0) \leq \max_{\alpha_0 \leq \alpha < 0.5} \left(c_\ell \alpha k_0 \log(\alpha k_0)) + c_\ell (1-\alpha) k_0 \log((1-\alpha)k_0)\right) + k_0.$$

It is easy to check that the maximum is achieved when $\alpha = \alpha_0$, and then

$$\tilde{\ell}(k_0) \leq \left(c_\ell \alpha_0 k_0 \log(\alpha_0 k_0)) + c_\ell (1-\alpha_0) k_0 \log((1-\alpha_0)k_0)\right) + k_0,$$

or,

$$\tilde{\ell}(k_0) \leq c_\ell k_0 \left(\alpha_0 \log(\alpha_0 k_0)) + (1-\alpha_0) \log((1-\alpha_0)k_0) + \frac{1}{c_\ell}\right) \tag{J.1}$$

$$= c_\ell k_0 \left(\alpha_0 (\log(\alpha_0) + \log k_0) + (1-\alpha_0)(\log(1-\alpha_0) + \log k_0) + \frac{1}{c_\ell}\right) \tag{J.2}$$

$$= c_\ell k_0 \left(\log k_0 + \alpha_0 (\log(\alpha_0)) + (1-\alpha_0)(\log(1-\alpha_0)) + \frac{1}{c_\ell}\right) \tag{J.3}$$

$$= c_\ell k_0 \log k_0 + c_\ell k_0 \left(\alpha_0 (\log(\alpha_0)) + (1-\alpha_0)(\log(1-\alpha_0)) + \frac{1}{c_\ell}\right). \tag{J.4}$$

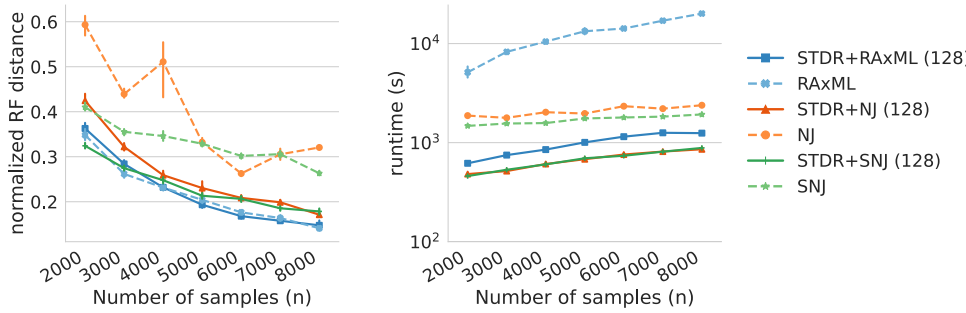


FIG. K5. Trees generated according to Kingman's coalescent model with $m = 2000$ terminal nodes. The mean and standard deviation of the normalized RF distance (left) between the reconstructed tree and the input tree and of the runtime (right) are shown for each method over five independent runs.

Since there is c_ℓ such that $\alpha_0(\log(\alpha_0)) + (1-\alpha_0)(\log(1-\alpha_0)) + \frac{1}{c_\ell} \leq 0$ we have that

$$\tilde{\ell}(k_0) \leq c_\ell k_0 \log k_0,$$

which concludes the proof. □

K. Additional Simulation Results

K.1 *Kingman's coalescent model*

We generated a random tree according to Kingman's coalescent model [33] with $m = 2000$ terminal nodes (see example in Fig A1). Figure K5 shows the accuracy (left panel) and the runtime (right panel) of the different methods as functions of the sequence length. The threshold parameter τ was set to 128 for all experiments. Here, STDR + RAxML performs similarly to RAxML in accuracy while achieving more than an order-of-magnitude reduction in runtime. Compared to NJ and SNJ, STDR + NJ and STDR + SNJ show improvement in both accuracy and runtime.

K.2 *Caterpillar tree*

We generated a caterpillar tree with $m = 512$ terminal nodes, where the non-terminal nodes form a path graph. The similarity between each pair of adjacent nodes was set to $\delta = 0.81$. As in Section 7, we compare NJ, SNJ and RAxML, with STDR where the aforementioned methods are used as subroutines. The STDR threshold is set to $\tau = 64$ for all three STDR variants. Figure K6 shows the normalized RF distance (left) and runtime (right) of the different methods as functions of the sequence length n. Here, all three methods are significantly improved when combined with STDR in both runtime and accuracy.

K.3 *Comparison to TreeMerge*

Here we compare the performance of STDR with RAxML as a subroutine to TreeMerge [39]. First, we generated random trees with 2000 terminal nodes according to the coalescent model. The trees were recursively partitioned by STDR with a threshold of $\tau = 128$. The structure of the different partitions was recovered by RAxML. We compared STDR's merging criteria with TreeMerge [39] for various sequence

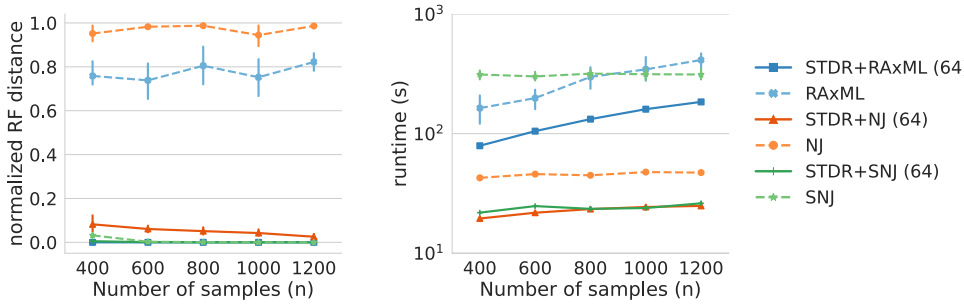


FIG. K6. A caterpillar tree with $m = 512$ terminal nodes. The mean and standard deviation of the runtime (right) and RF distance between the reconstructed tree and the input tree (left) are shown for each method over five independent runs.

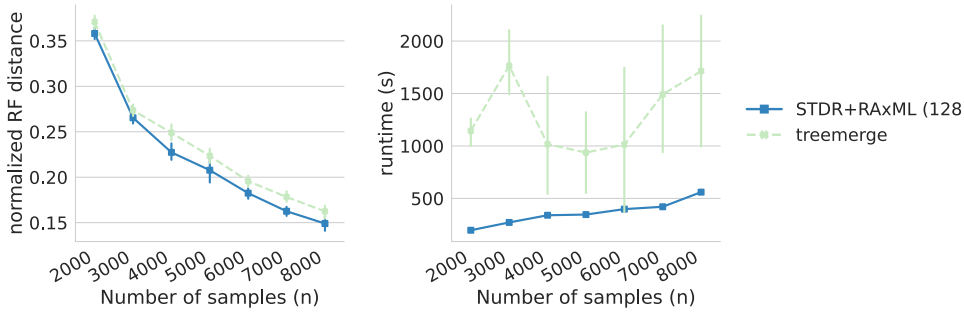


FIG. K7. A coalesent tree with $m = 2000$ terminal nodes. The mean and standard deviation of the normalized RF distance (left) between the reconstructed tree and the input tree and of the runtime (right) are shown for each method over five independent runs.

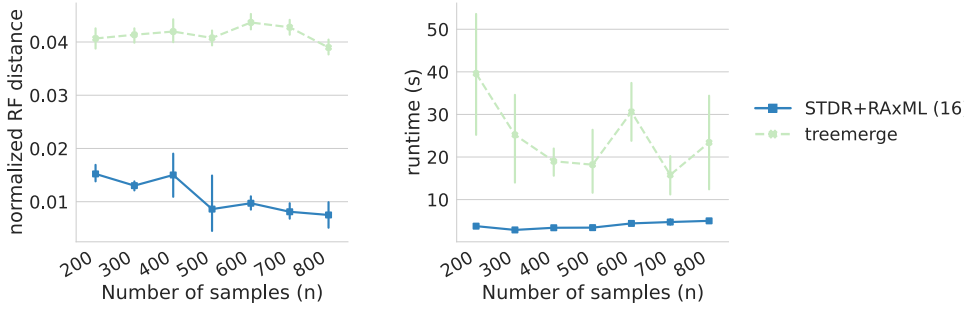


FIG. K8. A birth-death tree with $m = 100$ terminal nodes. The mean and standard deviation of the normalized RF distance (left) between the reconstructed tree and the input tree and of the runtime (right) are shown for each method over five independent runs.

lengths. The results are shown in Figure K7. The merging process of STDR achieved slightly better accuracy than TreeMerge, with a significantly reduced runtime. Next, we repeated the same experiment, but with a birth-death tree with birth rate of 0.5 and death rate of 0. We used $m = 100$ terminal nodes. Even for such small trees the merging of STDR improves over TreeMerge in both, reconstruction error and running time. The results are shown in Figure K8.