SCAMPP: Scaling Alignment-Based Phylogenetic Placement to Large Trees

Eleanor Wedell, Yirong Cai, and Tandy Warnow

Abstract—Phylogenetic placement, the problem of placing a "query" sequence into a precomputed phylogenetic "backbone" tree, is useful for constructing large trees, performing taxon identification of newly obtained sequences, and other applications. The most accurate current methods, such as pplacer and EPA-ng, are based on maximum likelihood and require that the query sequence be provided within a multiple sequence alignment that includes the leaf sequences in the backbone tree. This approach enables high accuracy but also makes these likelihood-based methods computationally intensive on large backbone trees, and can even lead to them failing when the backbone trees are very large (e.g., having 50,000 or more leaves). We present SCAMPP (SCaling AlignMent-based Phylogenetic Placement), a technique to extend the scalability of these likelihood-based placement methods to ultra-large backbone trees. We show that pplacer-SCAMPP and EPA-ng-SCAMPP both scale well to ultra-large backbone trees (even up to 200,000 leaves), with accuracy that improves on APPLES and APPLES-2, two recently developed fast phylogenetic placement methods that scale to ultra-large datasets. EPA-ng-SCAMPP and pplacer-SCAMPP are available at https://github.com/chry04/PLUSplacer.

Index Terms—Phylogenetic placement, maximum likelihood, phylogenetics, pplacer, EPA-ng

1 Introduction

Phylogenetic placement is the process of taking a sequence (called a "query sequence") and adding it into a phylogenetic tree (called the "backbone tree"). These methods are used for taxonomic identification, obtaining microbiome profiles, and biodiversity assessment [1], [2], [3], [4], [5], [6]. Furthermore, phylogenetic placement can be used to update very large phylogenies [7], where they offer a computationally feasible approach in comparison to *de novo* phylogeny estimation (which is NP-hard in most formulations). These two different applications of phylogenetic placement – taxonomic identification of reads in an environmental sample and large-scale phylogeny estimation – present different scalability challenges: the first requires the ability to process many reads, potentially millions, and the second requires the ability to add query sequences into increasingly large trees. In this paper, we mainly consider the scalability challenges in using phylogenetic placement to add query sequences into large trees.

Phylogenetic placement based on optimizing the maximum likelihood score is a natural approach, and is employed in pplacer [8], EPA [9], and EPA-ng [10] (an improved version of EPA). These likelihood-based phylogenetic placement methods have generally been found to have excellent accuracy but can be computationally intensive when placing into large trees, due to their use of likelihood calculations. Another limitation of likelihood-based placement methods is that they depend on multiple sequence alignments, which can reduce their applicability and also increase the computational effort in using the methods.

Other phylogenetic placement methods have been developed that enable potentially greater scalability and speed. RAPPAS [11] and App-SpaM [12] both focus on placement for unaligned query sequences. RAPPAS in particular shows promise in scalability to large backbones since it uses k-mers. However, both App-SpaM and RAPPAS were reported as being less accurate than pplacer [11], [12]. APPLES [7] is a distance-based approach to phylogenetic placement that has shown particularly good scalability, including to backbone trees with up to 200,000 sequences. APPLES-2 [13], a new version of APPLES, has subsequently been developed using a divide-and-conquer strategy to substantially improve upon the accuracy and speed of APPLES while maintaining its scalability. However, although APPLES and APPLES-2 can both scale to very large backbone trees, the maximum likelihood-based placement methods provide better accuracy on those datasets on which they can run [13].

Thus, maximum likelihood phylogenetic placement methods have accuracy advantages over alternative approaches, but many studies have restricted these methods, such as pplacer, to relatively small backbone trees due to a combination of reasons, including limitations in computational resources and potentially numeric issues (see further discussion in Supplementary Materials, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2022.3170386, Section S5). Of the other methods, APPLES-2 may be the most scalable and perhaps most accurate method, but has reduced accuracy compared to pplacer and has not been extensively studied. In particular, APPLES-2 has

• *The authors are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. E-mail: {ewedell2, yirongc2, warnow}@illinois.edu.*

Manuscript received 30 September 2021; revised 10 April 2022; accepted 22 April 2022. Date of publication 26 April 2022; date of current version 3 April 2023.
This work was supported in part by the U.S. National Science Foundation under Grants 1458652 and 2006069 (to TW), in part by Fellowship support from the Siebel Scholars Program (to EW), and in part by Grainger Foundation (to TW).
(Corresponding author: Eleanor Wedell.)
Digital Object Identifier no. 10.1109/TCBB.2022.3170386

not been examined for accuracy in placing fragmentary sequences.

To enable pplacer, EPA-ng, and other computationally intensive likelihood-based phylogenetic placement methods to be used on larger backbone trees, we have developed the SCAMPP (i.e., "SCaling AlignMent-based Phylogenetic Placement") framework, which we now describe. Rather than attempting to find the best location in the entire backbone tree into which we insert the query sequence, the SCAMPP framework uses an informed strategy to select a subtree of the backbone tree, places the query sequence into that subtree using the selected placement method, and then identifies the correct location in the backbone tree associated with that location. Using the SCAMPP framework with pplacer yields pplacer-SCAMPP and similarly, EPA-ng yields EPA-ng-SCAMPP, but any standard phylogenetic placement method that uses aligned sequences can be used within SCAMPP. The SCAMPP framework thus extends the provided phylogenetic placement method to enable it to scale to larger backbone trees and does not change the method when the backbone tree is small enough. This approach to phylogenetic placement focuses on placement locally within a subtree of the backbone tree, rather than searching the entire backbone tree for where to place the query sequence.

Our experimental study, using both biological and simulated datasets, shows that the SCAMPP framework enables pplacer and EPA-ng to be used with large backbone trees and maintains their accuracy on those datasets on which the placement methods can run, while reducing runtime and peak memory usage. A particular outcome of our study is that EPA-ng-SCAMPP and pplacer-SCAMPP can place into backbone trees with 200,000 leaves with accuracy that improves on APPLES-2, the prior leading method for phylogenetic placement on large backbone trees. Furthermore, although APPLES-2 remains generally the fastest of these methods, the difference in running time between pplacer-SCAMPP and APPLES-2 is relatively small on the largest datasets, and pplacer-SCAMPP is faster than APPLES-2 when placing fragmentary sequences into the largest backbone trees. Thus, the SCAMPP framework not only enables alignment-based phylogenetic placement methods to scale gracefully to large datasets, but its use with pplacer provides the best accuracy of all the existing phylogenetic placement methods we explore.

2 The SCAMPP Framework

2.1 Overview

The SCAMPP framework is designed to work with a provided phylogenetic placement method Φ, under the following basic assumptions about Φ. The input to Φ is (a) T, the "backbone tree", which is an unrooted binary tree with numeric parameters (including branch lengths) for its specified model of sequence evolution, (b) a set Q of query sequences, and (c) a multiple sequence alignment of the sequences at the leaves of the tree and the query sequences.

For each query sequence, Φ returns an output jplace file [14], consisting of multiple possible placement edges within the tree, each with a corresponding distal length, likelihood weight ratio, likelihood, and pendant branch length. The output can be used to identify a single edge into which the query sequence should be placed, as well as to produce support statistics about edge placements. The statistical support values are useful for metagenomic taxon identification and abundance profiling (e.g., as used in TIPP [3] and TIPP2 [4]). However, the output of the single best placement is also relevant when using phylogenetic placement for the purpose of incrementally building a large tree, as discussed in [7].

In this study, we focus on the use of phylogenetic placement to identify a single best edge within the backbone tree for a single query sequence; this is an application that can be used both for adding sequences into very large trees (e.g., incrementally building a gene tree) as well as for taxon identification.

Here we describe the SCAMPP framework for use with any given phylogenetic placement method Φ, when placing a single query sequence from Q; we note that inserting all the sequences in Q can be performed independently, and so this description will suffice to define the framework. We also describe the SCAMPP framework for the Generalized Time Reversible (GTR) [15] model for nucleotide evolution with gamma-distributed rates across sites, noting that modifications to this approach for other models (e.g., protein models) is trivial. The input to the SCAMPP framework has two algorithmic parameters, which are Φ (the phylogenetic placement method) and B, the maximum size for the placement subtree. The remaining parameters are the usual ones given to likelihood-based placement methods, and are:

- T, an unrooted tree with numeric substitution model (e.g., GTR) parameters (e.g., branch lengths, substitution rate matrix, stationary distribution, gamma distribution), with S the set of sequences labelling the leaves of T
- q: the query sequence to be inserted into T
- A: the multiple sequence alignment on $S \cup \{q\}$

When we use SCAMPP with Φ, we refer to the combination as Φ-SCAMPP; hence, EPA-ng-SCAMPP refers to using SCAMPP with the EPA-ng phylogenetic placement method, pplacer-SCAMPP refers to using SCAMPP with pplacer, etc. At a high level, our three-stage technique for Φ-SCAMPP operates as follows (see Fig. 1):

- Stage 1: A subtree T' of T is identified (defined by its set S' of leaves), with the restriction that T' cannot contain more than B leaves. This is referred to as the "placement tree".
- Stage 2: We apply Φ to T'; this returns a jplace file with the set of the edges selected by Φ for having good likelihood scores for the query sequence.
- Stage 3: For each edge e' in the jplace file, we find the associated edge e in T.

The output is therefore a jplace file containing all the potential placement edges and their associated likelihood scores. We study the SCAMPP framework in the context of finding the single best placement, but the output can be used more generally.

In what follows, we will assume that the backbone tree has n leaves and that the sequence alignment has length k.

2.2 Stage 1

The input to Stage 1 includes the value for B, which defines the size of the placement subtree, as well as the backbone

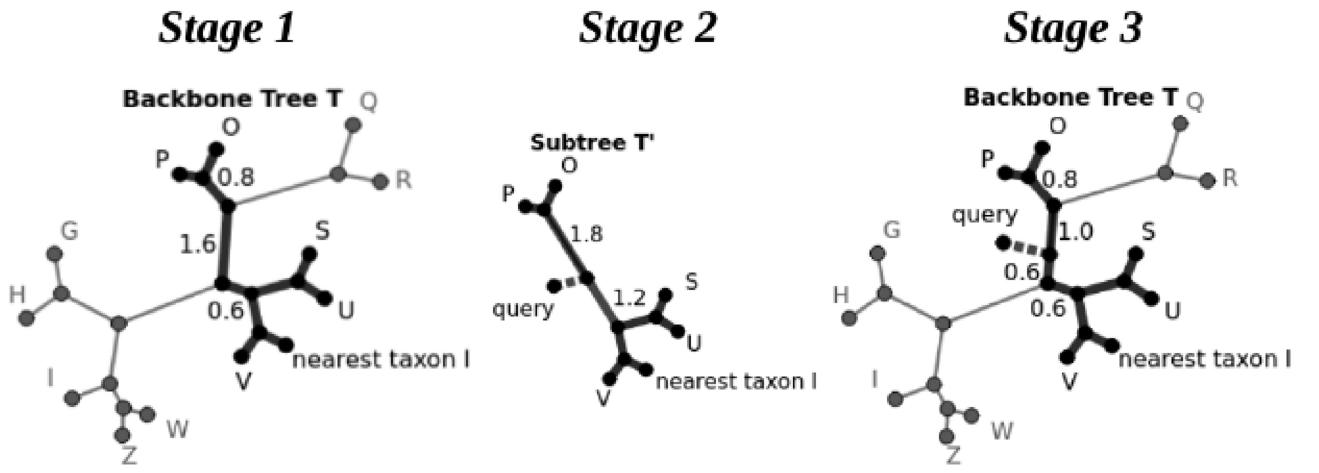


Fig. 1. Description of the SCAMPP technique. In Stage 1, we select the placement subtree T' from the backbone tree T, for a specified query sequence. To find the placement subtree T' of T, we first find the leaf l with the smallest Hamming distance to the query sequence (called the "nearest taxon"). Then, we greedily pick the $B-1$ leaves (here $B=6$) with the smallest distance to l. In this case, we select five leaves O,P,S,U,V, and the placement subtree T' is induced by the set $\{P,O,S,U,V,l\}$ of six leaves. Here we show the given placement method selecting an edge in T' separating leaves $\{P,O\}$ from $\{S,U,V,l\}$, and this single edge in T' corresponds to a path of three edges in T. Note that a viable phylogenetic placement method for the SCAMPP framework returns not only which edge in the placement subtree to insert the query sequence into, but the branch lengths on either side; this is used to find the correct placement of the query sequence in Stage 3.

tree T. Note that if the backbone tree is small enough (i.e., has at most B leaves), then the SCAMPP framework just defaults to the selected phylogenetic placement method; hence this algorithm only applies when the backbone tree has more than B leaves.

The first stage needs to select the subtree of B leaves into which to place the query sequence. The first step is to find a closest leaf l (defined by the Hamming distance, which is number of sites where the two sequences are different (i.e., both have different letters or one is gapped and the other is not). This is modified when the query sequence is identified as a fragment by the user, in which case the calculation is performed after removing the leading and trailing gaps. This calculation takes $O(nk)$ time. We call this closest leaf the "nearest taxon" to the query sequence.

Once the leaf l is found, we select the $B-1$ leaves in order of their path distance to l, as we now define. The path distance in T from a given leaf l' to l is $\sum_{e_i \in P} L(e_i)$ where P is the path in T from l to l' and $L(e_i)$ is the length of the edge e_i in P. Starting from l, we use a breadth-first search to select those leaves in T that have the lowest path distance to l until we select the $B-1$ additional leaves (thus forming the set of B leaves, after we add l). Once the set of B leaves is identified, the induced subtree T' is returned, with the branch lengths in T' computed by using the associated branch lengths in T (note that this subtree T' may not be a clade in T).

Stage 1 takes $O(nk)$ time, and returns a set of B leaves and the induced subtree T' (with its associated numeric parameters, induced on it by the backbone tree), which is the placement tree passed to a phylogenetic placement method in Stage 2.

2.3 Stage 2

We then run the given phylogenetic placement method on the placement tree T' we obtain from Stage 1. This identifies a collection of edges, each of which has a good likelihood score.

2.4 Stage 3

For each edge e' found in Stage 2, we find the single edge e in T corresponding to that edge. To do this, we first determine the set of edges in T that define the same bipartition as e'. This set will either be a single edge e or will define a path of two or more edges in T. Fig. 1 shows such an example of how an edge e' in the placement subtree T' corresponds to a path with more than a single edge from the given backbone tree T. We let $Path(e')$ denote the edge or path in T corresponding to e', noting that a single edge is also a path (albeit of length 1). To determine $Path(e')$ given e', note that e' defines a bipartition $\pi(e')$ on T'. At least one, and possibly more than one, of the edges in T define bipartitions that correspond to $\pi(e)$ (meaning specifically that they induce the same bipartition when restricted to the leafset of T'). The set of edges in T that define bipartitions corresponding to $\pi(e')$ form either a single edge or a path of two or more edges, and so defines $Path(e')$. We then set $L(e')$ (i.e., the length of edge e') to be $L(Path(e'))$, where $L(Path(e'))$ is the sum of the branch lengths in the path (or edge) in T denoted by $Path(e')$.

If e' corresponds to a single edge e in T, then we place the query sequence into that edge. However, if e' corresponds to a path with two or more edges in T, then we use the distances we obtained to find the correct placement edge for the query sequence, as we now describe and also show in Fig. 1.

Recall that the tree T' is a subtree of T formed by specifying a set of leaves, and that the edges of T' have branch lengths that correspond to the branch lengths in T. Recall also that when a phylogenetic placement method inserts the query sequence into e' in T', it also subdivides the edge e' and specifies how the branch length is divided. For example, suppose $e'=(a,b)$ is an edge in T' with length $L(e')$ and the query sequence is attached to this edge. Then the given phylogenetic placement method subdivides the edge e', thus creating two new edges (a,v) and (v,b), whose lengths add up to $L(e')$. We then use those new lengths to determine exactly what edge in T we should insert the query sequence into and

TABLE 1
Dataset Statistics

Dataset	number of sequences	alignment length	Type (bio or sim)	p-distance mean	p-distance maximum	gaps proportion
green85 [16]	5088	1486	biological	.250	.479	.146
LTP_s128_SSU [16]	12,953	1598	biological	.228	.468	.090
16S.B.ALL [17]	27,643	6857	biological	.210	.769	.118
nt78 [18]	78,132	1287	simulated	.404	.639	.006
RNASim [19]	200,000	1620	simulated	.410	.618	.051

The first column gives the name of the dataset and the publication describing the dataset. For each dataset we show the number of sequences, the length of the reference alignment, its type (biological or simulated), the mean and maximum p-distance (i.e., normalized Hamming distances) between pairs of sequences in the alignment, and the proportion of the alignment that is gapped.

where in that edge we should create a new node (to which we attach the query sequence) so as to produce the lengths specified by the phylogenetic placement method. An example of this is provided in Fig. 1, and another more complex example is provided in the Supplementary Materials, available online, Section S1 and Fig. S1.

3 Experimental Study

3.1 Overview

Recall that SCAMPP has two algorithmic parameters: the phylogenetic placement method Φ and the value for B, which is the maximum size (i.e., number of leaves) of the placement subtree. In our first experiment we explore how to set B within SCAMPP for use with Φ being either pplacer or EPA-ng. After selecting B, we use that value in all subsequent experiments. The second experiment compares pplacer-SCAMPP and EPA-ng-SCAMPP to other phylogenetic placement methods on backbone trees with up to 78,000 leaves, also for placing full-length sequences. The third experiment explores larger backbone trees with up to 200,000 leaves, and explores placement of full-length as well as fragmentary sequences. All methods were evaluated with respect to delta error [7], [20] (a measure of how much tree error increases by adding a query sequence) as well as running time and peak memory usage within a leave-one-out experiment.

All our analyses were performed in the same computational infrastructure (the Campus Cluster at the University of Illinois), which provides 64GB of memory, 18 CPUs, and up to 4 hours of runtime. Nevertheless, the machines vary in age and speed, and running times are not exactly comparable. Additional details of the methods, including version numbers and commands, are provided in the Supplementary Materials, available online, Section S4.

3.2 Methods

We evaluate EPA-ng-SCAMPP (v1.0.0) and pplacer-SCAMPP (v1.1.0) in comparison to APPLES (v1.1.3), APPLES-2 (v2.2.0), EPA-ng (v0.3.8), and pplacer (v1.1.alpha19).

3.3 Datasets

3.3.1 Overview

For our experiments we use five nucleotide datasets, with three biological and two simulated (Table 1). These datasets have alignments and reference trees (true alignments and true trees for the simulated datasets and estimated alignments and trees for the biological datasets) that range in size from roughly 5000 sequences to as large as 200,000 sequences. We use the two smallest datasets, both biological datasets from the PEWO collection [16], for Experiment 1, where we set the algorithmic parameter B (which determines the size of the subtree used in the SCAMPP framework); the other datasets are used in Experiments 2 and 3 for evaluating the impact of the SCAMPP framework. We also made versions of these datasets where the sequences are fragmentary. All datasets are available in public repositories, with locations provided at https://doi.org/10.13012/B2IDB-9257957_V1.

3.3.2 Biological Datasets

We use three biological datasets, two from the *PEWO* [16] collection and one from the Comparative Ribosomal Website [17]. The first PEWO dataset is the green85, originally from the Greengenes database [21], of 5088 aligned sequences and a reference tree that was computed on the alignment. The second PEWO dataset is LTP_s128_SSU, which contains 12,953 aligned sequences and a reference tree originally from [22], [23]. The final biological dataset is *16S.B.ALL*, which contains 27,643 sequences with an alignment based on secondary structure [17] and a RAxML [24] maximum likelihood tree.

3.3.3 Simulated Datasets

We use two collections of simulated datasets. The first is the *nt78* dataset, which contains 78,132 nucleotide sequences. This simulated dataset was created to evaluate the maximum likelihood method, FastTree 2 [25]. This dataset contains 20 simulated replicates, and we arbitrarily chose the first for this study. We generate an estimated tree for phylogenetic placement using FastTree 2 for this study.

The second collection comes from the *RNASim* dataset, which is a simulated dataset with ten replicates, each containing 1,000,000 sequences. The RNASim dataset is the result of a simulation where sequences evolve under a complex biophysical model that reflects selective pressures to maintain the RNA secondary structure. RNASim has been used in other studies to evaluate alignment accuracy [19], [26], [27], [28]. The RNASim Variable Size (RNASim-VS) datasets are subsets of varying sizes (up to 200,000 sequences), drawn at random from the million-sequence RNASim dataset. These RNASim-VS datasets were used in [13] to evaluate phylogenetic placement methods, and provide true phylogenetic tree, true multiple sequence alignment, and estimated maximum likelihood (ML) trees (obtained using FastTree 2 [25]) on each subset, which serve as the backbone trees. For each backbone tree size there are five replicates included in [7] (except for the largest which contains only one), and 200 query sequences per replicate.

3.3.4 Fragmentary Datasets

For Experiment 3, we created fragmentary versions of the RNASim datasets as follows. We created "low fragmentation" (LF) conditions where a quarter of the sequences are fragmentary (mean 25% of the original length, with a standard deviation of 60 nucleotides). We picked a random starting position within the randomly selected sequence, selected a random number L from a normal distribution with mean 25% the original length and standard deviation of 60, and extracted the next L nucleotides. "High fragmentation" (HF) conditions were also simulated in a similar manner, with a mean 10% of the original length with a standard deviation of 10 nucleotides. The resulting mean fragment length is 154 for the HF conditions and 385 for the LF conditions. The true alignments of resulting sequence fragments are used for placement.

3.3.5 Backbone Trees and Numeric Parameters

The phylogenetic placement methods need backbone trees with numeric parameters (branch lengths, substitution rate matrix, stationary distribution, and gamma distribution), with specific protocols for each method. It is recommended that APPLES-2 uses branch lengths estimated under minimum evolution [7], [13], and APPLES-2 will estimate these branch lengths using FastTree-2 prior to performing placement. In order to provide fair runtime analyses we provide APPLES with a tree estimated by FastTree-2 with the no ML option for all datasets. We used the minimum evolution branch lengths provided by [7] for use with APPLES and APPLES-2 on the RNASim-VS datasets, and for all other datasets we estimated these using FastTree-2 [18]. For EPA-ng and EPA-ng-SCAMPP, we re-estimated branch lengths for each estimated ML tree using RAxML-ng [29]. For pplacer-SCAMPP, on the RNAsim dataset we used trees with branch lengths estimated by FastTree 2 [25]. For pplacer-SCAMPP on all other datasets we estimated branch lengths using RAxML. The remaining parameters (i.e., 4×4 substitution rate matrix) across all datasets were estimated using RAxML [24] version 7. For pplacer we used the branch lengths and numeric parameters directly from RAxML version 7. However, pplacer failed to provide valid results on some large backbone trees using the numeric parameters produced by RAxML. Therefore, on those backbone trees where pplacer produced negative infinite likelihood scores using the default technique for numeric parameter estimation, we produced numeric parameters using an alternative technique recommended in [13]: we computed numerical parameters using FastTree 2 and then provided these parameters to taxit within Taxtastic [30]; this produced numeric parameters that we then used with pplacer. See Supplementary Materials, available online, Section S4 for additional information.

3.4 Leave-One-Out Study

Our leave-one-out evaluation operates as follows. Given a backbone tree on n leaves, a random leaf is selected and removed, thus producing a reduced tree on $n-1$ leaves. The sequence for that leaf is then added back into the reduced tree using the given phylogenetic placement method.

3.5 Criteria

We report running time, peak memory usage, and placement error. We report placement error by comparing trees, before and after a single query sequence is added to the backbone tree, to the true tree (when using simulated datasets) or a reliable estimated tree (when using biological data) on the corresponding set of leaves. We will refer to the true tree or reliable estimated tree as the "reference tree". This comparison is performed by representing each tree by its set of bipartitions, noting that each edge in a tree defines a bipartition on the leafset. We define the "false negative" error (also called the number of "missing branches") of a given tree t with respect to the reference tree to be number of edges (or bipartitions) that are in the reference tree but not in t. The change in the number of false negatives produced by placing a query sequence into a backbone tree is the "delta error" produced by the placement method.

We illustrate this calculation with an example. Suppose the backbone tree has leafset S and is missing 5 edges found in the true tree on this leafset. Now suppose we use a phylogenetic placement method to add a new sequence s' into the tree, so that the extended backbone tree now has leaves $S \cup \{s'\}$, and suppose this extended tree is missing 7 edges from the true tree on $S \cup \{s'\}$. Then the delta error is 2, since the number of edges that were missing went up by 2. Note that the delta error can never go down, since an edge that is missing before s' is added is still missing after s' is added.

We now make this concept precise using mathematical notation. Given a tree Y we let $B(Y)$ denote the set of bipartitions of Y. We let T^* denote the reference tree (i.e., either the true tree or a reliable estimated tree), and we assume T^* has leafset S. In a leave-one-out study, we are given a tree t and we delete one leaf s' from t, thus producing a tree T on leafset $S' = S \setminus \{s'\}$. Note that the tree t may not be the reference tree. When we add query sequence s' into T, we obtain a tree P. Note that P has the entire leafset S. We let $T^*|_{S'}$ denote the subtree of T^* induced by leafset S'. Then the delta error for P, denoted by $\Delta_e(P)$, is given by the following formula:

$$\Delta e(P) = |B(T^*) \setminus B(P)| - |B(T^*|_{S'}) \setminus B(T)|, \tag{1}$$

where $|X|$ denotes the number of elements in the set X. Thus the first term is the number of false negatives for the tree P and second term is the number of false negatives for the tree T. Note that $\Delta_e(P) \geq 0$, since the number of missing branches produced by adding a query sequence to a tree cannot decrease.

We note that several earlier studies [10], [16], [31] have used the "node distance" to evaluate phylogenetic placement methods within leave-out studies, as follows. A starting tree is given and a leaf is deleted, and then reinserted using a placement method. The distance (i.e., number of nodes) from the final placement to the placement in the starting tree is the node distance. However, this is equal to the delta error when the starting tree is interpreted as the true tree. Therefore, the delta error is an extension of the node distance that allows error in the starting tree (which can be quantified in a simulation study) to be part of the evaluation. Therefore, throughout our experiments, we use the delta error for both simulated and biological datasets, with the trees provided for the biological datasets treated as reference trees. (In other words, when we report delta error on the biological datasets, it is the same as reporting node distance for these datasets.)

4 Results

Results for Experiments 1–3 are shown here for APPLES-2, EPA-ng, pplacer, EPA-ng-SCAMPP, and pplacer-SCAMPP. APPLES was clearly inferior to APPLES-2 with respect to runtime, memory usage, and accuracy, and the results for APPLES are thus provided only in the Supplementary Materials, available online, Section S2 and Fig. S2. Results shown are restricted to those replicates where all methods returned valid outputs.

4.1 Experiment 1: Evaluating the Impact of the Parameter B on the SCAMPP Framework

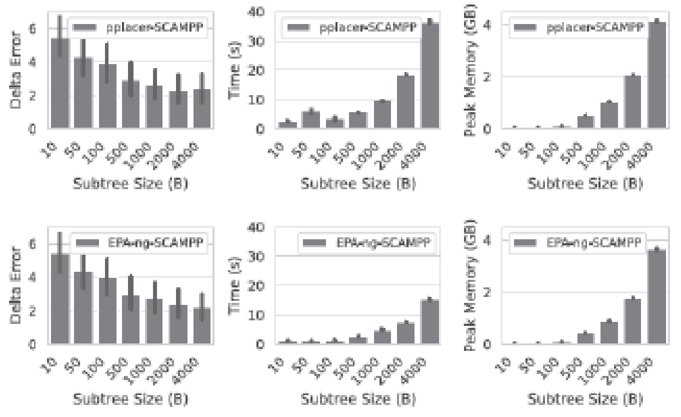
An important algorithmic parameter is the size B of the subtree into which the query sequence is placed, and exploring this question is the focus of this section. Previous studies have suggested that better placement accuracy is obtained by placing into a larger subtree [3], [20], indicating that the best placements may be obtained by increasing the value of B, which limits the placement subtree size given to the phylogenetic placement method. However, increasing the placement subtree size too much also increases the computational effort, and may also lead to failures in some cases.

In order to understand the impact of B, which determines the placement subtree size, we used two relatively small PEWO datasets (green85 with 5088 sequences and LTP_s128_SSU with 12,953 sequences) and tested a range of subtree sizes. We see (Fig. 2) that small values for B (which limit the placement to small subtrees) produced high error, but values for B generally between 1000 and 4000 had good accuracy (with very small differences between $B = 1000$ and $B = 4000$). However, as subtree sizes increased, runtime and memory usage also increased. Based on these trends, we performed a more focused evaluation of settings for B in the range between 1000 and 4000.

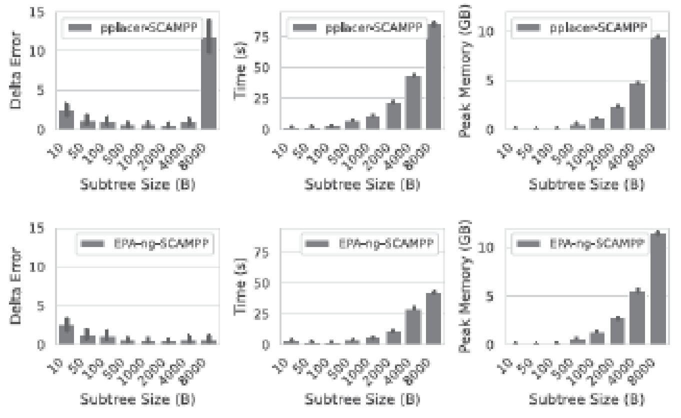
In Fig. 3, we compare pplacer-SCAMPP and EPA-ng-SCAMPP to the other phylogenetic placement methods using these three values for B to get a sense for how important it was to set B optimally. We examine the impact on placement error first, and then the impact on runtime and memory usage.

On the smaller of these two datasets (i.e., green85, with only 5088 sequences), we see that EPA-ng is slightly more accurate than EPA-ng-SCAMPP when $B = 1000$ or $B = 2000$ and then matches accuracy when $B = 4000$. This suggests that EPA-ng is able to provide a good analysis of the full dataset and that $B = 4000$ is slightly better than $B = 2000$. In contrast, for all settings of B, pplacer is clearly less accurate than pplacer-SCAMPP, and there is little difference between pplacer-SCAMPP for $B = 2000$ and $B = 4000$. This suggests that pplacer is unable to provide good accuracy on the full dataset and benefits from restriction to a subtree, and that $B = 2000$ is as good as $B = 4000$. Results on the larger of the two datasets (i.e., LTP_s128_SSU) are somewhat different than on the smaller dataset. First, all methods except for APPLES-2 have very low placement error, with average delta error below 1. In addition, there are very small differences the remaining methods, and changes to B on this dataset does not have much impact. Across the two datasets, setting $B = 2000$ or $B = 4000$ are both reasonable settings, with $B = 2000$ somewhat better for pplacer-SCAMPP and $B = 4000$ somewhat better for EPA-ng-SCAMPP.

A comparison of running time provides additional insights about how to set B. Specifically, increasing B increases the running time for both pplacer-SCAMPP and EPA-ng-SCAMPP, and has a larger impact on pplacer-SCAMPP than on EPA-ng-SCAMPP. In addition, pplacer has by far the highest running time (see for example runtime on the LTP_s128_SSU dataset) and EPA-ng is in second page, but setting $B = 2000$ in pplacer-SCAMPP or EPA-ng-SCAMPP greatly reduces the runtime. Furthermore, changing B from 2000 to 4000 approximately doubles the runtime



(a) green85 (5088 sequences)



(b) LTP_s128_SSU (12,953 sequences)

Fig. 2. Experiment 1: Exploring the impact of how B is set, which specifies the placement subtree size, on two PEWO biological datasets (LTP_s128_SSU and green85). Within each row the subfigures for pplacer-SCAMPP and EPA-ng-SCAMPP show: Mean delta error (left), Mean time in seconds (center), and Mean peak memory usage in Gb (right). Rows (from top to bottom) show results on green85 and LTP_s128_SSU. Results are shown across 200 query sequence placements.

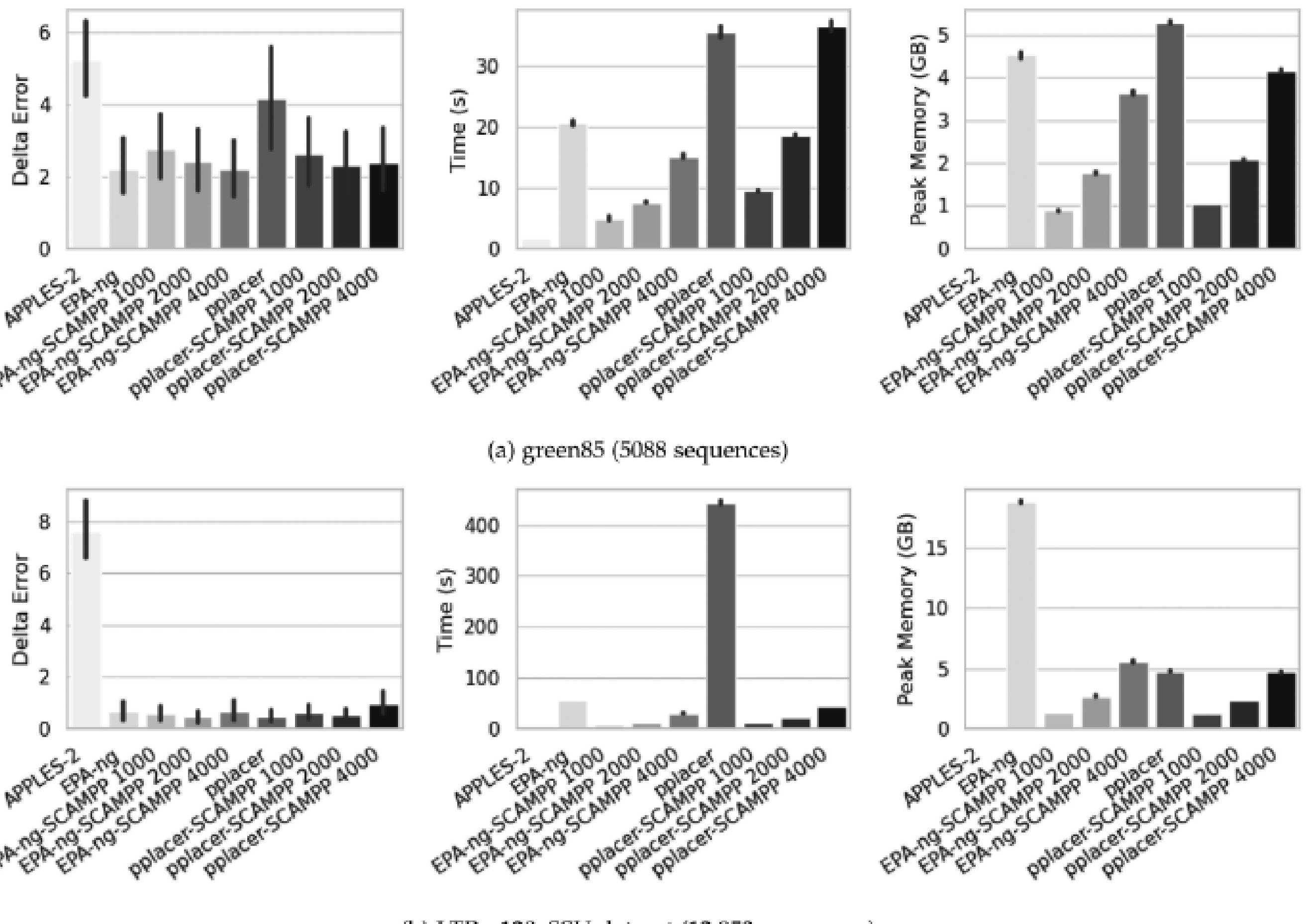


Fig. 3. Experiment 1: Results of phylogenetic placement methods on the two PEWO datasets (green85 and LTP_s128_SSU), using $B = 1000$, $B = 2000$, and $B = 4000$. Within each row the subfigures show: Mean delta error (left), Mean time in seconds (center), and Mean peak memory usage in Gb (right). Results are shown across 200 query sequence placements.

for both pplacer-SCAMPP and EPA-ng-SCAMPP. Thus, B has a large impact on runtime, as expected.

The peak memory usage by pplacer-SCAMPP and EPA-ng-SCAMPP is also very impacted by the setting for B. On the green85 dataset, the lowest peak memory usage is achieved by both methods when $B = 1000$, and then increases substantially with increases in B. The highest peak memory usage is for pplacer, followed by EPA-ng in second place, and every explored setting for B reduces their peak memory usage. On the LTP_s128_SSU dataset, the same trends appear, but with the following difference: here, EPA-ng has by far the highest peak memory usage (more than three times that of every other method).

Overall, what these trends show is that the different settings for B between 1000 and 4000 result in at worst small changes to the placement error but very large changes to runtime and memory usage. If placement accuracy must be optimized, then these results suggest that the optimal setting for B when using pplacer-SCAMPP is probably 2000, but the optimal setting when using EPA-ng-SCAMPP is possibly $B = 4000$ (but $B = 2000$ produces very close results). However, the computational hit (both running time and memory usage) in changing from $B = 2000$ to $B = 4000$ is substantial for both methods. Based on these experimental results, we set $B = 2000$ for default usage with both pplacer-SCAMPP and EPA-ng-SCAMPP, and used this setting in the subsequent experiments.

4.2 Experiment 2: Evaluating SCAMPP on Moderately Large Trees

This experiment examines phylogenetic placement on two moderately large backbone trees, using the setting for B established in Experiment 1. We analyze the 16S.B.ALL biological dataset (with 27,643 sequences) and the nt78 simulated dataset (with 78,132 sequences). We see different trends for each of these datasets, and so we discuss them separately, starting with the smaller of the two datasets.

On the 16S.B.ALL dataset (Fig. 4a), we see that APPLES-2 has double the delta error of the other methods. The most accurate method is EPA-ng, with delta error of 4.4, but the delta errors of the remaining methods are all between 5.3 and 5.4, and have overlapping error bars with EPA-ng. There are substantial differences in terms of running time, with APPLES-2 by far the fastest and pplacer by far the slowest. EPA-ng is the second slowest. In contrast, pplacer-SCAMPP and EPA-ng-SCAMPP (which are nearly identical in running time) are nearly as fast as APPLES-2. The methods also differ substantially with respect to memory usage, with APPLES-2 the best, followed closely by pplacer-

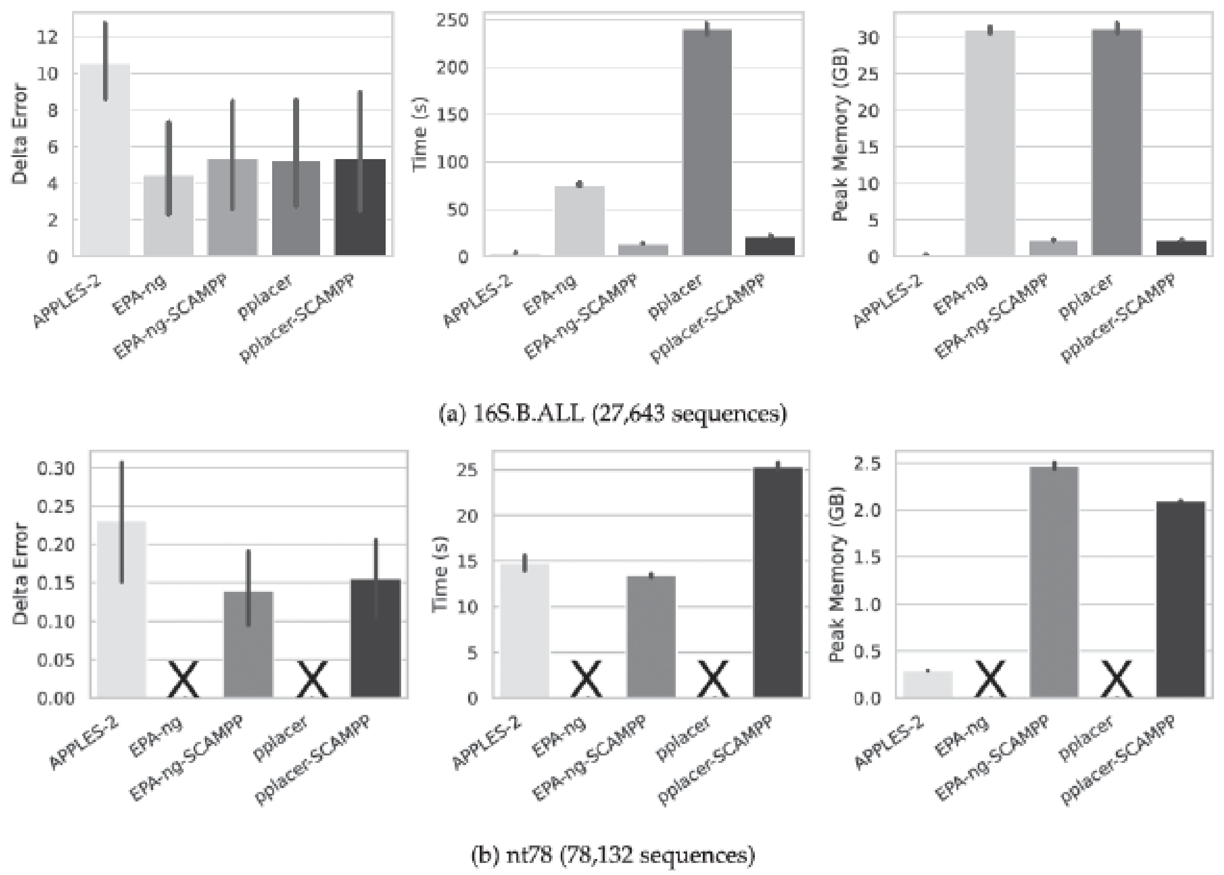


Fig. 4. Experiment 2: Results of phylogenetic placement methods on moderately large backbone trees. Within each row the subfigures show: Mean delta error (left), Mean time in seconds (center), and Mean peak memory usage in Gb (right). Rows (from top to bottom) show results on 16S.B.ALL and nt78. On the nt78 dataset, pplacer and EPA-ng because both fail to return valid results, and this is indicated with an "X". Results are shown across 200 query sequence placements.

SCAMPP and EPA-ng-SCAMPP, and then by EPA-ng and pplacer, which have about the same (large) memory usage. Specifically, SCAMPP enables a large reduction in peak memory usage for both pplacer and EPA-ng, from over 30Gb to under 3Gb on this dataset.

Results on the 78nt dataset (Fig. 4b) show somewhat different trends. The first and most significant difference is that neither pplacer nor EPA-ng were able to perform the placements. On this dataset both pplacer and EPA-ng failed to return a jplace file due to segmentation faults (see Supplementary Materials, available online, Section S5). The comparison between the remaining methods shows APPLES-2 less accurate than EPA-ng-SCAMPP and pplacer-SCAMPP, and with a small advantage to EPA-ng-SCAMPP. The three methods are again distinguishable in terms of runtime and memory usage, with APPLES-2 the fastest and using the least memory. A comparison between EPA-ng-SCAMPP and pplacer-SCAMPP shows pplacer-SCAMPP slower than EPA-ng-SCAMPP but using less memory.

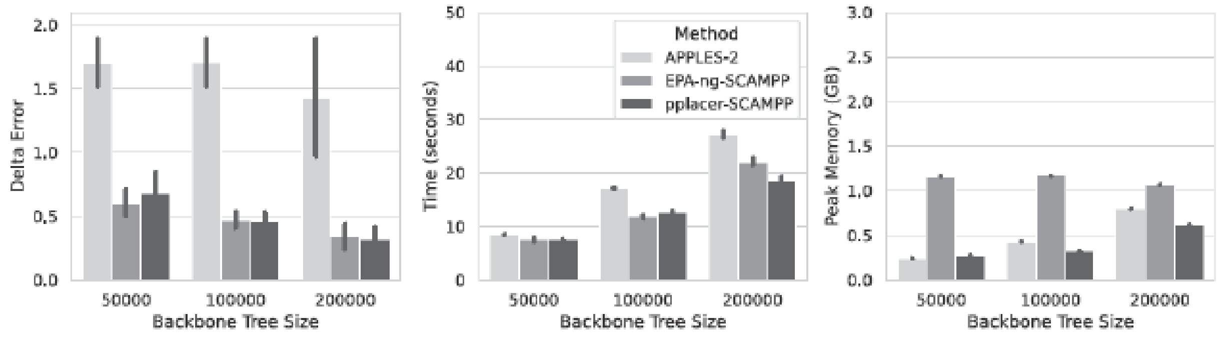
4.3 Experiment 3: Evaluating SCAMPP on Very Large Trees

Here we explore performance of phylogenetic placement methods when the backbone trees are very large, using the RNASim-VS datasets with backbone trees ranging from 50K to 200K leaves. There are five replicates each for trees with 50K and 100K leaves and only one replicate with a tree of 200K leaves. For this study, we do not use either pplacer or EPA-ng, as they fail to complete on the nt78 dataset, as shown in Experiment 2.

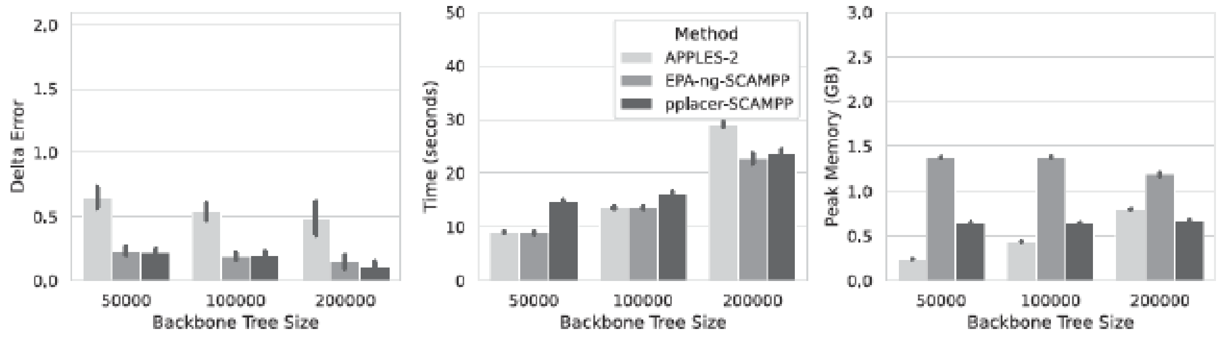
4.3.1 *Experiment 3a: Scalability in Placing Full-Length Sequences*

Placement error results on these data present interesting trends (Fig. 5c). On the backbone trees with 50,000 leaves, pplacer-SCAMPP has the lowest placement error, followed by EPA-ng-SCAMPP, and then by APPLES-2. On the 100,000-leaf backbone trees, pplacer-SCAMPP again has the lowest error, and APPLES-2 and EPA-ng-SCAMPP have the same higher error. Results on the 200,000-leaf backbone tree show the same relative trends as on the 100,000-leaf backbone trees, but error rates have dropped somewhat for all methods.

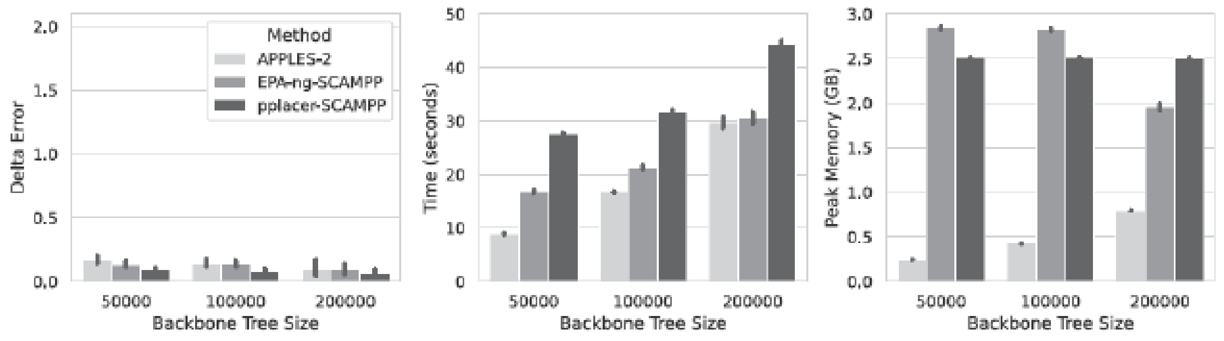
A comparison of methods with respect to running time and memory usage is also interesting (Fig. 5c). APPLES-2 is clearly the fastest of the three methods, followed by EPA-ng-SCAMPP and then by pplacer-SCAMPP. Furthermore, EPA-ng-SCAMPP and APPLES-2 are not very far apart in terms of runtime on the 100,000-leaf tree and then identical in running time on the largest tree with 200,000 leaves. Memory usage also clearly favors APPLES-2, and the differences between EPA-ng-SCAMPP and pplacer-SCAMPP are very small.



(a) Short fragments (average length 154)



(b) Long fragments (average length 385)



(c) Full-length sequences (average length 1539)

Fig. 5. Experiment 3: Comparison of placement of full length and fragmentary sequences on the RNASim-VS backbone trees with 50,000 to 200,000 leaves. Short fragments (top row), long fragments (middle row), and full-length sequences (bottom row). Within each row, we show placement delta error (left), runtime (center), and peak memory usage (right). We report results for 200 queries over 5 replicates on the 50,000- and 100,000-taxon backbone trees and only one replicate on the 200,000-taxon backbone trees. Each replicate contains 200 query sequence placements.

4.3.2 *Experiment 3b: Scalability of Fragmentary Sequence Placement*

We examined two lengths for the fragmentary sequences: short fragments, averaging 154 nucleotides, and slightly longer fragments, averaging 385 nucleotides. We refer to the shorter sequence condition as HF (high fragmentary) and the slightly longer fragments as LF (low fragmentary).

Results on the short fragments show very clear trends (Fig. 5a). First, APPLES-2 has substantially higher delta error than pplacer-SCAMPP and EPA-ng-SCAMPP for all backbone tree sizes, and the difference between pplacer-SCAMPP and EPA-ng-SCAMPP is very small. All three methods have essentially the same running time for backbone tree size 50,000 but differences appear as the backbone tree size increases so that APPLES-2 becomes the slowest of the three methods. EPA-ng-SCAMPP and pplacer-SCAMPP have close runtimes, with pplacer-SCAMPP slightly slower than EPA-ng-SCAMPP on backbone tree size 100,000 and then faster on backbone tree size 200,000. APPLES-2 and pplacer-SCAMPP both have relatively low peak memory usage at all sizes (though APPLES-2 uses more peak memory than pplacer-SCAMPP on the larger backbone trees), and EPA-ng-SCAMPP has by far the highest peak memory usage.

Results on the longer fragments show very similar trends, but with a few differences (Fig. 5b). As with the short fragments, APPLES-2 is the least accurate, and differences between pplacer-SCAMPP and EPA-ng-SCAMPP are minor (though there is a small advantage to pplacer-SCAMPP over EPA-ng-SCAMPP on the largest backbone size). The same basic trends hold for running time, except that pplacer-SCAMPP is the slowest of the three methods until the largest backbone size, where it is faster than

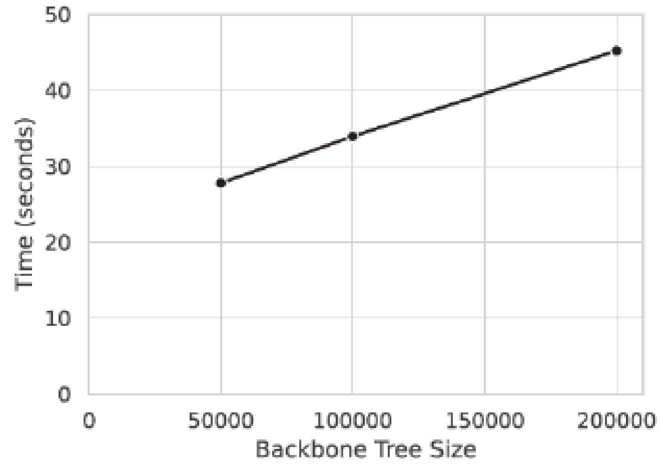


Fig. 6. Mean running time for pplacer-SCAMPP on three different RNA-sim backbone tree sizes, 50,000, 100,000, and 200,000, in placing full-length sequences. We report results for 5 replicates on the 50,000- and 100,000-taxon backbone trees and only one replicate on the 200,000-taxon backbone trees. Each replicate contains 200 query sequence placements.

APPLES-2 but slightly slower than EPA-ng-SCAMPP. Peak memory usage is also slightly different, but EPA-ng-SCAMPP is still by far the most memory-intensive.

Some other trends are also worth noting. First, delta error rates drop with increases in the backbone tree size, while runtime increases. The increase in runtime is expected, but the decrease in delta error is surprising, and worth further investigation. Interestingly, peak memory usage is fairly constant as the backbone tree size increase for EPA-ng-SCAMPP, but grows for pplacer-SCAMPP and for APPLES. The impact of backbone tree size on peak memory usage for APPLES-2 follows from its algorithm design, but the differential impact on pplacer-SCAMPP and EPA-ng-SCAMPP is somewhat surprising and worth further investigation. We also see that error rates are higher on the short fragments than on the long fragments, which is also expected (since there is less information available for placement).

4.3.3 *Computational Scalability of pplacer-SCAMPP*

We finish this section with a direct evaluation of how well pplacer-SCAMPP scales in terms of runtime and memory usage, by comparing runtime and memory usage on the RNASim-VS datasets for both full-length and fragmentary sequences. The runtime of pplacer-SCAMPP is close to linear in the size of the backbone tree (Fig. 6), and a detailed evaluation of the runtime of each step (Table 2) shows that the time used by pplacer itself is constant across all backbone tree sizes. We similarly see that the peak memory usage does not increase with backbone tree size (Fig. 5), suggesting that the maximum likelihood phylogenetic placement method is likely the process where the memory usage peaks, because the placement tree within the backbone tree remains a fixed size within our experiments. Overall, the computational scalability of pplacer-SCAMPP is very good on these datasets, and suggests the potential for being scalable to even larger backbone trees.

5 Discussion

5.1 Impact of Using SCAMPP

In general, we find that SCAMPP improves computational performance (both memory usage and running time) for both phylogenetic placement methods, but the impact on accuracy depends on the model condition and even depends on whether pplacer or EPA-ng is used. We also see that the relative accuracy and computational efficiency (both speed and memory) compared to a leading fast phylogenetic placement method, APPLES-2, depends on the model condition. We therefore begin with a discussion of these trends.

Starting with a comparison between pplacer and pplacer-SCAMPP, we note that on those datasets on which pplacer could run, pplacer-SCAMPP was always at least as accurate as pplacer, often substantially more accurate, and was also faster and had a smaller peak memory usage. We also see that the subtree size, as defined by B, has a large impact on pplacer: when B is very small (e.g., below 500 on the datasets in Experiment 1), delta error rates are high, then error rates drop as B increases to (approximately) 2000, but as B increases beyond that the error rates can become very high. This is most noteworthy on the LTP_s128_SSU dataset, where there is a very large increase in error at $B = 8000$. This trend shows that pplacer accuracy degrades on very large placement trees, which is an intriguing finding. Since we also observed that pplacer can return $-\infty$ values on large backbone trees (Supplementary Materials, available online, Section S5 and Table S1), this suggests that the issue is numerical. Taking these observations together, we infer that pplacer has numerical issues that make it not work that well (in terms of accuracy) on large placement trees, which is why the use of SCAMPP improves accuracy. In other words, the parameter B in the pplacer-SCAMPP pipeline limits the size of the subtree into which pplacer is applied, which eliminates (or at least reduces, depending on B) this numerical instability. The advantage of pplacer-SCAMPP over pplacer for accuracy, however, is not an inherent aspect of pplacer, since addressing the numerical instability of pplacer in a future implementation of pplacer may eliminate the accuracy advantage (or even reverse it). Thus, the main longterm advantage of pplacer-SCAMPP over pplacer may be computational performance.

The comparison between EPA-ng and EPA-ng-SCAMPP presents somewhat different trends. On those datasets on which both methods run, we sometimes see EPA-ng more accurate and sometimes less accurate than EPA-ng-SCAMPP; while these differences are small, the fact that EPA-ng-SCAMPP can be less accurate than EPA-ng indicates a noteworthy difference between EPA-ng and pplacer. We also see that increases in B beyond 2000 has a small but occasionally negative imapct on EPA-ng-SCAMPP. On the other hand, EPA-ng can fail to run on some datasets due to memory requirements. Overall, these trends suggest that EPA-ng may not have the same numeric vulnerability as we saw in pplacer (see Supplementary Materials, available online, Section S5 for additional discussion about this issue). In sum, our study shows that EPA-ng can provide good accuracy on those large backbone trees on which it can run, and the benefit to using EPA-ng within the SCAMPP

TABLE 2
Runtime Breakdown for pplacer-SCAMPP on Full-Length Sequences

RNASim Tree Size	Time Per Process in Seconds					
	Loading Data	Finding Nearest Taxon	Extracting Subtree	Running pplacer	Backbone Placement	Total Runtime
50,000	3.24	3.88	0.37	20.21	0.06	27.76
100,000	6.64	6.43	0.48	20.29	0.07	33.91
200,000	13.37	10.60	0.80	20.33	0.09	45.19

framework may only be to enable it to run within the available computational resources.

SCAMPP has a larger beneficial impact, especially for runtime and memory usage, for placing fragmentary sequences than it does for full-length sequences, which is also interesting. Specifically, pplacer-SCAMPP and EPA-ng-SCAMPP become much more computationally efficient on fragmentary sequences compared to full-length sequences, while APPLES-2 does not become more efficient on fragmentary sequences. These reductions in runtime and memory use are likely due to the masking techniques for leading and trailing gaps used in the SCAMPP framework as well as in EPA-ng and pplacer [10].

Overall, therefore, using SCAMPP greatly reduced runtime and memory usage for both pplacer and EPA-ng, and either improved accuracy or at worst slightly decreased accuracy; however, the decreases in accuracy were limited to EPA-ng-SCAMPP. Moreover, there were many large backbone trees on which neither pplacer nor EPA-ng could run, and using SCAMPP enabled them to run and with low memory usage. Thus, there was always a benefit obtained in using SCAMPP in our experiments, but the type of benefit and its magnitude depends on the placement method, backbone tree, and query sequence length.

5.2 Choosing Between Phylogenetic Placement Methods

This study establishes that pplacer-SCAMPP, EPA-ng-SCAMPP, and APPLES-2 can be used on large backbone trees, and that pplacer and EPA-ng are not as scalable as these three methods. Here we discuss the question of which of the three scalable methods should be used, and under which conditions. A comparison between pplacer-SCAMPP and EPA-ng-SCAMPP shows little difference in terms of accuracy, and differences for runtime and memory usage that depend on the dataset. Here we examine the relative performance of pplacer-SCAMPP and APPLES-2, as an example of when this framework can be useful.

Throughout the datasets we studied, pplacer-SCAMPP was more accurate than APPLES-2, but the degree of improvement varied. With the exception of the RNASim-VS datasets with fragmentary sequences, APPLES-2 was faster and had lower peak memory usage than pplacer-SCAMPP; however, the gap narrowed for the largest backbone trees. Thus, the choice between the two methods is fundamentally a tradeoff between computational performance (generally but not always favoring APPLES-2) and accuracy (favoring pplacer-SCAMPP). When accuracy is very important, the accuracy advantage may be large enough to merit the extra computational hit in using pplacer-SCAMPP instead of APPLES-2. However, the large difference in runtime under many conditions suggests that for some applications (most notably for microbiome analyses of millions of reads), APPLES-2 will be the method of choice.

5.3 Related Studies

This work builds off of an earlier prototype [32], which was limited to the SCAMPP framework's use with pplacer, and only explored a single model condition (RNAsim VS) with only full-length sequences. In the Supplementary Materials, available online, Section S3, we compare that implementation of SCAMPP to the current implementation, demonstrating that the new implementation is much faster and has lower memory usage that the initial implementation.

There are two relatively closed related approaches to the SCAMPP framework that require discussion: pplacerDC [33] and the multilevel "Russian Doll" phylogenetic placement technique [34]. We discuss each in turn.

The pplacerDC [33] technique employs a more exhaustive approach than pplacer-SCAMPP, but also enables pplacer to scale to larger backbone trees. In pplacerDC, the backbone tree is divided (through edge deletions) into placement subtrees with a bounded number of leaves, and then the query sequence is placed into each of the placement subtrees using pplacer. Each such placement is then embedded in the full backbone tree, thus producing an extended backbone tree, and the maximum likelihood score is estimated (using the provided numeric parameters, which are not re-optimized) using RAxML. The extended backbone trees thus have all the taxa (including the query sequence) and so can be compared to each other with respect to their maximum likelihood scores. The tree with the best likelihood score is then returned.

The study evaluating pplacerDC was limited to the RNASim VS datasets, where it was shown to be able to scale to backbone trees with 100,000 leaves. However, pplacerDC failed on backbone trees 200,000 leaves and was extremely computationally intensive on those datasets on which it completed – with higher running time and peak memory usage than pplacer-SCAMPP. Finally, pplacerDC was not evaluated on datasets with fragmentary sequences. Since both methods have been run on the same datsets (RNASim-VS with full-length sequences), the comparison shows that pplacerDC is slower, has higher delta error (e.g., about twice as high on the 100,000-leaf backbone tree), and higher peak memory usage. To understand the differences in runtime and peak memory usage, recall that pplacerDC requires that the query sequence be placed into all the placement subtrees created by the decomposition, which makes the runtime increase linearly with the backbone tree size (unless run with unbounded parallelism). There is also a memory and runtime hit produced by the use of RAxML to compute the likelihood score of each extended tree (and the number of these trees grows linearly with the size of the backbone tree). This discussion shows that by design pplacerDC is more computationally intensive than pplacer-SCAMPP, and that this additional expense will be seen on any dataset, not just the RNASim-VS datasets.

Another useful strategy for addressing the limited scalability of phylogenetic placement methods with respect to the backbone tree size is the multilevel placement method [34] that is also available within the GAPPA suite of tools [35]. The multilevel "Russian Doll" placement approach is described for use with a taxonomy (on a carefully selected set of sequences), but the general technique can be extended for use with any rooted tree. A sparse but representative subset of leaves from the rooted tree is selected, and is then used as the backbone tree (where it is referred to as a "broad backbone tree" (BT)). A phylogenetic placement method is then used to place the query sequence into the BT, which allows it to identify the best clades (rooted subtrees) for more careful exploration. The query sequence can then be placed in each of the clades, and the best placement(s) for each query sequence can be identified. By design, this multi-stage process reduces the need to place into the full backbone tree, and so reduces the computational effort for phylogenetic placement. The approach is very different from ours in a few ways, but the main difference is that it requires rooted trees. Nevertheless, it is a very interesting approach, and extending this technique to work with unrooted trees merits investigation.

5.4 Other Future Work

In previous sections we have identified some directions for future work. Here we discuss additional opportunities for developing the approach we have described here, as well as alternative approaches.

Improving the SCAMPP Design. The current SCAMPP strategy places a query sequence by finding a nearest taxon (i.e., a leaf that has minimum Hamming distance to the query sequence) and then extracts a subtree with B leaves using that leaf. Thus, the current strategy has only one algorithmic parameter (B) beyond the choice of the placement method. Our default is $B = 2000$, but Experiment 1 and our additional evaluations reported in the Supplementary Materials, available online, Section S6 and Fig. S4) suggested the possibility that the optimal setting for B might depend both on the dataset properties and on the phylogenetic placement method. In particular, if accuracy is the most important objective, then it seems possible that larger values of B might improve accuracy for EPA-ng-SCAMPP, and that very small values might suffice for sufficiently "easy" datasets. Hence a better understanding of the impact of dataset properties on this parameter selection is needed. Moreover, our placement subtree construction approach is very simple, and it is possible that other techniques for extracting a placement subtree might provide improved accuracy compared to this technique, even if the runtime and memory usage does not change.

We also note that EPA-ng has been optimized for placing large numbers of query sequences into backbone trees. This is an advantage for EPA-ng that is not enabled in the SCAMPP framework, which gives the placement method a different subtree for each query sequence. In order to take advantage of the batch processing offered by methods such as EPA-ng, a different divide-and-conquer framework would need to be explored.

Additional Evaluation. More generally, a full evaluation of the SCAMPP phylogenetic placement approach requires additional study. We performed a leave-one-out study, but a more extensive analysis including leave-clade-out study should be explored. We also did not explore the impact of alignment error in the phylogenetic placement pipeline, and so this should also be examined. Finally, we explored pplacer-SCAMPP and EPA-ng-SCAMPP in the context of growing a large tree, but they should also be evaluated for use in microbiome abundance profiling and taxon identification, as some of the most accurate such methods use phylogenetic placement. Thus, there are several directions for future work that have the potential to lead to improved understanding of how to design phylogenetic placement methods for use in different downstream applications.

6 Conclusion

Phylogenetic placement is a basic computational step in several bioinformatics pipelines, including incremental construction of very large phylogenies and taxonomic identification of reads obtained in metagenomic analyses. Of the many phylogenetic placement methods that have been developed, methods that use maximum likelihood, such as pplacer and EPA-ng, has been found to be the most accurate. Unfortunately, these likelihood-based methods are difficult to use with moderately large backbone trees (i.e., they can fail to return valid outputs or may have excessive memory requirements), which has meant that other phylogenetic placement methods are necessary. Methods such as APPLES-2 use distance-based techniques to perform phylogenetic placement, and are particularly fast and scalable; however, this and other studies have shown distance-based placement to not provide the same level of accuracy as likelihood-based methods.

We have presented the SCAMPP framework, a three-stage procedure for scaling alignment-based phylogenetic placement methods. We evaluated the SCAMPP framework for use with two such methods, pplacer and EPA-ng. Our study showed that using SCAMPP allowed both pplacer and EPA-ng to scale to backbone trees with 200,000 leaves without high peak memory requirements, thus greatly surpassing the limitations of these methods when used outside the framework. For those datasets on which pplacer could run, we also saw that pplacer-SCAMPP had better accuracy than pplacer, was faster, and used less peak memory. While EPA-ng-SCAMPP was sometimes less accurate than EPA-ng, those reductions in accuracy tended to be small and the improvement in running time and peak memory usage was very high. Thus, in general SCAMPP provides computational benefits to both methods and either improves accuracy (for pplacer) or has a variable impact (for EPA-ng) that tends to be minor.

One of the interesting trends we saw in this study is that although pplacer-SCAMPP improved on APPLES-2 for accuracy in all the cases we evaluated, the differences in some cases were extremely small; furthermore, in nearly all cases, APPLES-2 was the fastest and least memory-intensive method. Thus, it is not at all obvious that any one method dominates the others. This is particularly important, given that computational performance may be a limiting factor, making it by necessity a requirement to use the fastest method, or the least memory-intensive method, on a given dataset. In considering the different factors that impact accuracy and runtime/memory usage, we suggest that APPLES-2 be used when highly accurate phylogenetic placement seems likely, as it tends to be the most computationally efficient, but that pplacer-SCAMPP or EPA-ng-SCAMPP be used under other conditions. In particular, there may be a benefit to using pplacer-SCAMPP or EPA-ng-SCAMPP instead of APPLES-2 when the query sequences are short, as is typical in metagenomic datasets.

Acknowledgments

The authors wish to thank the three anonymous reviewers for their substantive and detailed suggestions and critiques, which led to improvements in the study and manuscript.

References

[1] S. Conlan, H. H. Kong, and J. A. Segre, "Species-level analysis of DNA sequence data from the NIH Human Microbiome project," *PLoS One*, vol. 7, no. 20, 2012, Art. no. e47075.

[2] C. O. McCoy and F. A. Matsen, "Abundance-weighted phylogenetic diversity measures distinguish microbial community states and are robust to sampling depth," *PeerJ*, vol. 1, 2013, Art. no. e157.

[3] N.-P. Nguyen, S. Mirarab, B. Liu, M. Pop, and T. Warnow, "TIPP: Taxonomic identification and phylogenetic profiling," *Bioinformatics*, vol. 30, no. 24, pp. 3548–3555, 2014.

[4] N. Shah, E. K. Molloy, M. Pop, and T. Warnow, "TIPP2: Metagenomic taxonomic profiling using phylogenetic markers," *Bioinformatics*, vol. 37, pp. 1839–1845, 2021.

[5] P.-A. Chaumeil, A. J. Mussig, P. Hugenholtz, and D. H. Parks, "GTDB-Tk: A toolkit to classify genomes with the Genome taxonomy database," *Bioinformatics*, vol. 36, no. 6, pp. 1925–1927, 2020.

[6] H. M. Bik, D. L. Porazinska, S. Creer, J. G. Caporaso, R. Knight, and W. K. Thomas, "Sequencing our way towards understanding global eukaryotic biodiversity," *Trends Ecol. Evol.*, vol. 27, no. 4, pp. 233–243, 2012.

[7] M. Balaban, S. Sarmashghi, and S. Mirarab, "APPLES: Scalable distance-based phylogenetic placement with or without alignments," *Syst. Biol.*, vol. 69, no. 3, pp. 566–578, 2020.

[8] F. A. Matsen, R. B. Kodner, and E. V. Armbrust, "pplacer: Linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree," *BMC Bioinf.*, vol. 11, no. 1, 2010, Art. no. 538.

[9] S. A. Berger, D. Krompass, and A. Stamatakis, "Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood," *Syst. Biol.*, vol. 60, no. 3, pp. 291–302, 2011.

[10] P. Barbera *et al.*, "EPA-ng: Massively parallel evolutionary placement of genetic sequences," *Syst. Biol.*, vol. 68, no. 2, pp. 365–369, 2019.

[11] B. Linard, K. Swenson, and F. Pardi, "Rapid alignment-free phylogenetic identification of metagenomic sequences," *Bioinformatics*, vol. 35, no. 18, pp. 3303–3312, 2019.

[12] M. Blanke and B. Morgenstern, "App-SpaM: Phylogenetic placement of short reads without sequence alignment," *Bioinf. Adv.*, vol. 1, no. 1, 2021, Art. no. vbab027.

[13] M. Balaban, Y. Jiang, D. Roush, Q. Zhu, and S. Mirarab, "Fast and accurate distance-based phylogenetic placement using divide and conquer," *Mol. Ecol. Resour.*, vol. 22, pp. 1213–1227, 2022.

[14] F. A. Matsen, N. G. Hoffman, A. Gallagher, and A. Stamatakis, "A format for phylogenetic placements," *PLoS One*, vol. 7, no. 2, 2012, Art. no. e31009.

[15] S. Tavaré, "Some probabilistic and statistical problems in the analysis of DNA sequences," *Lectures Math. Life Sci.*, vol. 17, no. 2, pp. 57–86, 1986.

[16] B. Linard, N. Romashchenko, F. Pardi, and E. Rivals, "PEWO: A collection of workflows to benchmark phylogenetic placement," *Bioinformatics*, vol. 36, no. 21, pp. 5264–5266, Jul. 2020.

[17] J. J. Cannone *et al.*, "The comparative RNA web (CRW) site: An online database of comparative sequence and structure information for Ribosomal, Intron, and other RNAs," *BMC Bioinf.*, vol. 3, no. 1, 2002, Art. no. 2.

[18] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree 2–Approximately maximum-likelihood trees for large alignments," *PLoS One*, vol. 5, no. 3, 2010, Art. no. e9490.

[19] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, "PASTA: Ultra-large multiple sequence alignment for nucleotide and amino-acid sequences," *J. Comput. Biol.*, vol. 22, no. 5, pp. 377–386, 2015.

[20] S. Mirarab, N. Nguyen, and T. Warnow, "SEPP: SATé-enabled phylogenetic placement," in *Proc. Pacific Symp. Biocomput.*, 2012, pp. 247–258.

[21] T. Z. DeSantis *et al.*, "Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB," *Appl. Environ. Microbiol.*, vol. 72, no. 7, pp. 5069–5072, 2006.

[22] P. Yarza *et al.*, "The all-species living tree project: A 16S rRNA-based phylogenetic tree of all sequenced type strains," *Syst. Appl. Microbiol.*, vol. 31, no. 4, pp. 241–250, 2008.

[23] P. Yarza *et al.*, "Update of the all-species living tree project based on 16S and 23S rRNA sequence analyses," *Syst. Appl. Microbiol.*, vol. 33, no. 6, pp. 291–299, 2010.

[24] A. Stamatakis, "RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models," *Bioinformatics*, vol. 22, no. 21, pp. 2688–2690, 2006.

[25] M. N. Price, P. S. Dehal, and A. P. Arkin, "FastTree 2–Approximately maximum-likelihood trees for large alignments," *PLoS One*, vol. 5, no. 3, 2010, Art. no. e9490.

[26] N. D. Nguyen, S. Mirarab, K. Kumar, and T. Warnow, "Ultra-large alignments using phylogeny-aware profiles," *Genome Biol.*, vol. 16, no. 1, pp. 1–15, 2015.

[27] V. Smirnov and T. Warnow, "Phylogeny estimation given sequence length heterogeneity," *Syst. Biol.*, vol. 70, no. 2, pp. 268–282, 2021.

[28] V. Smirnov and T. Warnow, "MAGUS: Multiple sequence alignment using graph clustering," *Bioinformatics*, vol. 37, pp. 1666–1672, 2021.

[29] A. M. Kozlov, D. Darriba, T. Flouri, B. Morel, and A. Stamatakis, "RAxML-NG: A fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference," *Bioinformatics*, vol. 35, no. 21, pp. 4453–4455, 2019.

[30] FHCRC, "Taxtastic software, version 9.9.2," distributed by Fred Hutchison Computational Biology, 2022. Accessed: Feb. 20, 2022. [Online]. Available: https://github.com/fhcrc/taxtastic

[31] F. A. Matsen, R. B. Kodner, and E. V. Armbrust, "pplacer: Linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree," *BMC Bioinf.*, vol. 11, no. 1, 2010, Art. no. 538.

[32] E. Wedell, Y. Cai, and T. Warnow, "Scalable and accurate phylogenetic placement using pplacer-XR," in *Proc. Int. Conf. Algorithms Comput. Biol.*, 2021, pp. 94–105, https://link.springer.com/chapter/10.1007/978-3-030-74432-8_7

[33] E. Koning, M. Phillips, and T. Warnow, "pplacerDC: A new scalable phylogenetic placement method," in *Proc. 12th ACM Conf. Bioinf. Comput. Biol. Health Informat.*, 2021, pp. 1–9.

[34] L. Czech, P. Barbera, and A. Stamatakis, "Methods for automatic reference trees and multilevel phylogenetic placement," *Bioinformatics*, vol. 35, no. 7, pp. 1151–1158, 2019.

[35] L. Czech, P. Barbera, and A. Stamatakis, "Genesis and Gappa: Processing, analyzing and visualizing phylogenetic (placement) data," *Bioinformatics*, vol. 36, no. 10, pp. 3263–3265, 2020.

Eleanor Wedell received the undergraduate degree in applied mathematics from Drexel University, Philadelphia, Pennsylvania, in 2015. She is currently working toward the MS degree with Tandy Warnow in the Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, Illinois. In 2021, she was selected as a Siebel Scholar for her research in computational biology and scientometrics. She received Honorable Mention in the 2022 National Science Foundation Graduate Research Fellowship competition. She will begin her PhD in the Department of Computer Science at the University of Illinois Urbana-Champaign in Fall 2022.

Yirong Cai received the bachelors' degree from the University of Illinois Urbana-Champaign, Champaign, Illinois, in May 2021, where she majored in mathematics and computer science. She is currently working toward the PhD degree with the Computer Science Department, McGill University, Montreal, Canada. Her current research is focused on astrocyte geometry.

Tandy Warnow received the PhD degree in mathematics from UC Berkeley, Berkeley, California, in 1991 under the direction of Gene Lawler. She is the Grainger Distinguished Chair in Engineering with the Department of Computer Science, University of Illinois Urbana-Champaign. Her research focuses on reconstructing complex and large-scale evolutionary histories. She was awarded the David and Lucile Packard Foundation Award (1996), a Radcliffe Institute Fellowship (2003), and the John Simon Guggenheim Foundation Fellowship (2011). She was elected fellow of the of the Association for Computing Machinery (ACM) in 2015, of the International Society for Computational Biology (ISCB) in 2017, and of the American Association for the Advancement of Science (AAAS) in 2021.