Scalable and Accurate Phylogenetic Placement Using pplacer-XR

Eleanor Wedell(✉), Yirong Cai, and Tandy Warnow(✉)

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA
{ewedell2,yirong2,warnow}@illinois.edu

Abstract. Phylogenetic placement, the problem of placing a 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 method, pplacer, performs the placement using maximum likelihood but fails frequently on backbone trees with 5000 sequences. We show a simple technique, pplacer-XR (pplacer-eXtra Range), that extends pplacer to large datasets. We show, using challenging large datasets, that pplacer-XR provides the accuracy of pplacer and the scalability to ultra-large datasets of a leading fast phylogenetic placmement method, APPLES. pplacer-XR is available in open source form on github.

Keywords: Phylogenetic placement · Likelihood-based methods · Phylogenetics

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 [5–7,9,12,14]. Furthermore, phylogenetic placement can be used to update very large phylogenies [2], where they offer a computationally feasible approach in comparison to *de novo* phylogeny estimation (which is NP-hard in most formulations).

One of the earliest methods for phylogenetic placement is pplacer [8], which uses a maximum likelihood approach (as well as a Bayesian method) to add query sequences into the tree. The input is a query sequence and a backbone tree, as well as a multiple sequence alignment of the sequences at the leaves of the backbone tree as well as the query sequence. The backbone tree is also given with numeric parameters (branch lengths, substitution rate matrix, etc.) for the selected sequence evolution model (e.g., the Generalized Time Reversible model [16]). Given this input, pplacer finds the best edge (under the maximum

Y. Cai and E. Wedell—Contributed equally to this work.

C. Martín-Vide et al. (Eds.): AlCoB 2021, LNBI 12715, pp. 94–105, 2021.
https://doi.org/10.1007/978-3-030-74432-8_7

likelihood criterion) in the backbone tree to attach the query sequence to, then subdivides the edge and returns the enlarged tree as well as the updated branch lengths, and a confidence score for the placements returned. EPA [4] was developed in 2011 with the same likelihood-based approach and accuracy, and was replaced in 2019 by EPA-ng [3], an improved version of EPA. In 2020, Balaban et al. [2] developed APPLES, a distance-based approach to phylogenetic placement. Balaban et al. evaluated EPA-ng, pplacer, and APPLES on a range of model conditions and revealed that while EPA-ng and pplacer provide high accuracy (with pplacer more accurate than EPA-ng), only APPLES was able to run on large backbone trees. Furthermore, pplacer, although the most accurate of the three methods, was limited to backbone trees with only a few thousand sequences, and often failed in [2] when the backbone tree had 5000 sequences. When pplacer failed in [2], these occurred when pplacer was able to finish running but produced placements with confidence scores of $-\infty$ (negative infinity). We are also aware that pplacer has high memory requirements, as reported in the github site for GTDB-tk [6]. Thus, pplacer, currently the most accurate placement method available, may be limited to small backbone trees (unless, perhaps, there is access to large amounts of memory), while APPLES, the most scalable method, has reduced accuracy compared to both EPA-ng and pplacer.

To scale pplacer to larger datasets, we use the following strategy, which we call "pplacer-XR" (pplacer-eXtra Range). Rather than attempting to find the best location in the entire backbone tree into which we insert the query sequence, pplacer-XR uses an informed strategy to select a subtree of the backbone tree, places the query sequence into that subtree, and then identifies the correct location in the backbone tree associated with that location. The first and third of these steps involve using distances, and the middle step uses pplacer; hence pplacer-XR is an extension of pplacer to enable it to run on larger backbone trees (and is identical to pplacer on those backbone trees that are small enough for pplacer to run). This approach is unique amongst placement methods, in that it focuses on placement locally among a subset of taxa, rather than the performing statistical analysis on the entire set of sequences. Our experimental study, using the datasets from the study presenting APPLES [2], shows that pplacer-XR, although slower than APPLES, improves on the accuracy of APPLES and matches its scalability. Our study also shows that pplacer-XR is more accurate and also faster than EPA-ng (which, like pplacer, does not scale to large backbone trees). Thus, pplacer-XR is the most accurate of the existing phylogenetic placement methods, and matches the scalability (but not the speed) of the most scalable of these methods.

The remainder of the paper is organized as follows. In Sect. 2 we present the pplacer-XR method. We describe the experimental study in Sect. 3. The results of the study are presented in Sect. 4, and we close with a discussion in Sect. 5.

2 pplacer-XR

The input to pplacer-XR is (1) a backbone tree T with a set S of sequences labelling the leaves, (2) a query sequence q, (3) a multiple sequence alignment A

(of length k) that includes all of the sequences in S as well as the query sequence q, and (4) a parameter B (to be the largest size that pplacer can reliably run on). We describe pplacer-XR for a single query sequence, as placing a set of sequences can be done independently. We also assume that the backbone tree is provided with the numeric parameters (branch lengths, substitution rate matrix, etc., as computed using RAxML [15]), associated with the specified sequence evolution model. However, if the backbone tree does not have these numeric parameters, we use RAxML to estimate the parameters (and will need to resolve the tree if it is not binary before using RAxML). We assume that pplacer-XR is used to add nucleotide sequences into a backbone tree under the GTR [16] model, but we note that other models could be used without any important change to the method. At a high level, our three-stage technique operates as follows:

- 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 sequences (and we set $B = 2000$ by default). This is referred to as the "placement tree".
- Stage 2: We use pplacer to find the best edge e' of T' for the query sequence.
- Stage 3: Since e' may correspond to a path with more than one edge in T, we use distances (estimated by pplacer) to find the correct edge of T into which we add the query sequence.

Stage 1: A key issue in Stage 1 is how to set B, which limits the size of the placement tree. In general, larger placement trees provide better accuracy (as shown in [11]), but also run the risk of pplacer failing (due to the backbone tree being too large) and will increase the running time. Hence, we have set $B = 2000$, which is a size that pplacer reliably completes on. Note that if the backbone tree is small enough (i.e., has at most B leaves), then pplacer-XR defaults to pplacer; hence this algorithm only applies when the backbone tree has more than B leaves.

In order to obtain a subtree most likely to contain the correct placement, pplacer-XR uses a greedy strategy to find a set of B leaves in T that are close to the query sequence. The first step is to find a closest leaf l (using the Hamming distance to the query sequence, which is well defined since we are given a multiple sequence alignment). Hence, this step takes $O(nk)$ time.

Once that leaf l is found, pplacer-XR uses a breadth first search to find $B-1$ additional leaves, here based on the path length to the leaf l, defined as follows. 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 reach B leaves, and we return the subtree of T induced by this set of leaves. 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. Hence, Stage 1 takes $O(nk)$ time, and returns a set of B leaves and the induced subtree T' (with its associated branch lengths and other numeric parameters), which will be the placement tree passed to pplacer in Stage 2.

Stage 2: We then run pplacer on the placement tree T' we obtain from Stage 1. This identifies an edge e' in T', which will correspond to either a single edge e or a path $P(e')$ of two or more edges in T. Since a single edge is vacuously a path, we will let $P(e')$ denote the edge or path in T corresponding to e'. To determine $P(e')$ 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 are compatible with $\pi(e)$ (meaning specifically that they induce the same bipartition when restricted to the leafset of T'). The set of edges in T that define bipartitions compatible with $\pi(e')$ form either a single edge or a path, and defines $P(e')$. We then set $L((e')$ (i.e., the length of edge e') to be $L(P(e'))$, where $L(P(e'))$ is the sum of the branch lengths in the path (or edge) in T denoted by $P(e')$.

Stage 3: 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.

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. Specifically, if an edge in T' also exists in T, then they have the same branch length, and if the edge e' in T' corresponds to a path P in T, then the length of the edge e' is the sum of the lengths of the edges in P. Recall also that when pplacer 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 pplacer 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 how to divide the length of that edge to produce the desired outcome.

3 Experimental Study Design

Overview. We used simulated datasets from [2] to explore pplacer-XR in comparison to other phylogenetic placement methods. In these analyses, the available sequences were evolved down a model tree and backbone trees were constructed on these sequences. Thus, query sequences (also generated in the simulation) can be added into the backbone trees using phylogenetic placement methods. The true tree on every subset of the sequences is therefore known, and so error produced by a phylogenetic placement method can be exactly quantified. We used the leave-one-out approach from [2] to evaluate accuracy. Specifically, Balaban et al. [2] provided 200 query sequences for each model condition they studied. Given query sequence q, we extract the leaf for q from the initial backbone tree, thus producing the reduced backbone tree which we denote by T. We then use the placement method to insert q into T, thus producing a tree with the same set of leaves as the initial backbone tree.

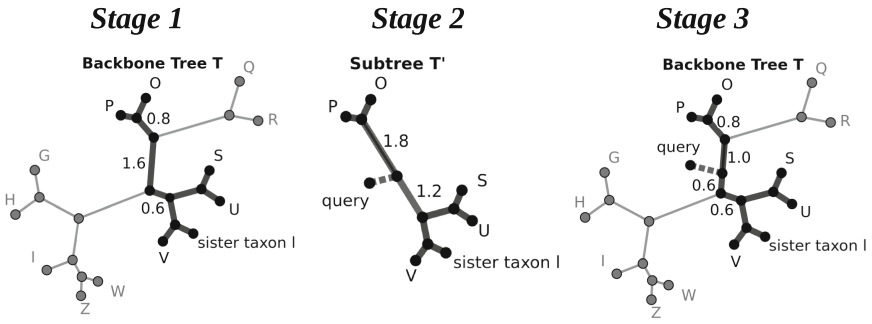

Fig. 1. Description of the pplacer-XR technique. In Stage 1, we select the placement subtree T' from the backbone tree T, for a specified query sequence. To find the placement subtree T' of T, we first find the leaf l with the smallest Hamming distance to the query sequence (here called the "sister taxon"). Then, we greedily pick the $B-1$ leaves (here $B=6$) that have the smallest path distance to l (using branch lengths to define path distances). 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 pplacer 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 pplacer 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.

Because pplacer has already been established in [2] to be the most accurate method when it can run and because pplacer and pplacer-XR are the same when pplacer can run, we only explore results on datasets that are too big for pplacer. Therefore, we selected the larger backbone trees from the study by Balaban et al. [2] to evaluate pplacer-XR. Specifically, we examine a set of "variable size" backbone trees with 5000 to 200,000 leaves, each a subset of the RNASim [10] million-sequence simulated dataset (which evolved under a complex biophysical model that includes selection). The backbone trees for these datasets were estimated using FastTree2 on the RNASim alignments.

We compare pplacer-XR to EPA-ng v0.3.8 using the default settings under the GTR model [16] with the Γ model of rate variation [17], and APPLES v1.3.0 run using its default settings. We ran all analyses on the University of Illinois Campus Cluster, which limits all analyses to four hours. Therefore, any analyses that did not complete within four hours were marked as "failures".

Backbone Trees and Branch Lengths. We use the RNASim variable size datasets studied in Balaban et al. [2], with the same backbone trees (topologies and branch lengths), alignments, and query sequences. These variable size datasets have backbone trees computed on the true alignment using FastTree2 [13] and contain 5,000, 10,000, 50,000 and 100,000 leaves (each with five replicates) and 200,000 (one replicate). We use the query sequences provided in [2] for each backbone tree; each such tree has 200 query sequences, which are drawn from the backbone

tree, and thus are in the original RNAsim dataset. The true alignment for the RNASim dataset, modified to remove the sites with more than 95% gaps, is provided to all phylogenetic placement methods. Balaban et al. computed branch lengths on all the backbone trees using protocols specific to each placement method (e.g., FastME for APPLES and RAxML for EPA-ng and pplacer); we use their backbone trees and branch lengths for EPA-ng and APPLES, and use the same command for RAxML to compute branch lengths for pplacer-XR as they used for pplacer.

Performance Criteria. We report placement error using the "delta error", as used by Balaban et al. [2] (described below). We also report the running time, recognizing that since the methods were run on a heterogeneous system (the University of Illinois Campus Cluster), these are not exactly comparable.

The delta error is defined using the concept of "false negatives", which we now define. Note that every edge in an unrooted tree t defines a bipartition on its leafset, and the set $B(t)$ of these bipartitions defines the tree t. Now let T^* denote the true tree on n leaves and let T denote a tree estimated on a subset $\mathcal{L}$ of $n-1$ of the leaves of T^*. We can restrict T^* to the leaves $\mathcal{L}$ of T, and denote this tree by $T^*|_{\mathcal{L}}$. Then, the *false negatives* of the estimated tree T with respect to the true tree T^* are the bipartitions in $B(T^*|_{\mathcal{L}}) \setminus B(T)$. When we insert a query sequence into tree T we obtain a tree P with one additional leaf (and hence the same number of leaves as T^*); the number of false negatives can therefore go up or stay the same, but cannot decrease. This means that the delta error is always non-negative. It is, however, possible for the delta error to be 0 without any of the estimated trees being correct (e.g., imagine the case where the backbone tree is not identical to the true tree, but the query sequence is correctly placed, perhaps as the sibling to another leaf; in such a case, the delta error will be 0 although the backbone tree before and after placement has errors). Put formally, the delta error produced by adding the missing query sequence into the backbone tree T (which is lacking one leaf) to obtain tree P is

$$\Delta e(P) = |B(T^*) \backslash B(P)| - |B(T^*|_{\mathcal{L}}) \backslash B(T)|. \quad (1)$$

We report the average and maximum delta error as well as the number of placements with a delta error of 0.

4 Results

All three methods were run on all datasets. APPLES and pplacer-XR successfully completed on all datasets, including the analyses with backbone trees containing 200,000 leaves. However, results with EPA-ng on backbone trees of size 50,000 were not obtained for just over 800 placements due to a core dump while running. This seemed to be a result of memory limitations, rather than time limitations. As a result, for the backbone trees larger than 10,000 taxa pplacer-XR

is compared only to APPLES. All other placement methods (except EPA-ng) were able to complete within the 4-hr time limit.

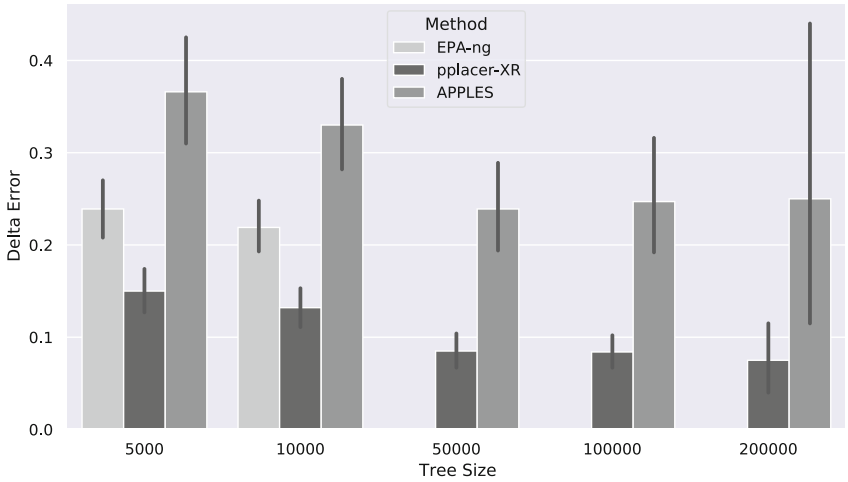

Fig. 2. Delta error (means plus standard error) over of all placements for each method over five backbone tree sizes. Averages for the 5,000, 10,000, 50,000, and 100,000 taxa trees are over all 1,000 query sequence placements, and for the 200,000 taxa tree the averages are for 200 query sequence placements. EPA-ng was unable to run for trees larger than 10,000 taxa and so no results are reported for EPA-ng on those datasets. This was run on the UIUC Campus Cluster with 64 GB of memory per placement and 4 h time limit.

4.1 Query Placement Accuracy

Figure 2 and Table 2 show that the mean delta error was significantly lower for the pplacer-XR than for APPLES and EPA-ng. EPA-ng was more accurate than APPLES on those datasets on which it completed, but could only reliably complete on the 5000- and 10,000-sequence backbone trees. The delta error results indicate that for all methods the accuracy improves as the backbone tree size increases, suggesting that denser taxon sampling improves phylogenetic placement accuracy. However, for every backbone tree size, APPLES has 2–3 times the mean delta error of pplacer-XR.

Figure 3 and Table 3 show the maximum delta error in each dataset, which corresponds to the placement with the highest delta error in that set. Note that pplacer-XR has lower maximum delta error than EPA-ng and APPLES, and APPLES has the highest maximum error (specifically, pplacer-XR has maximum

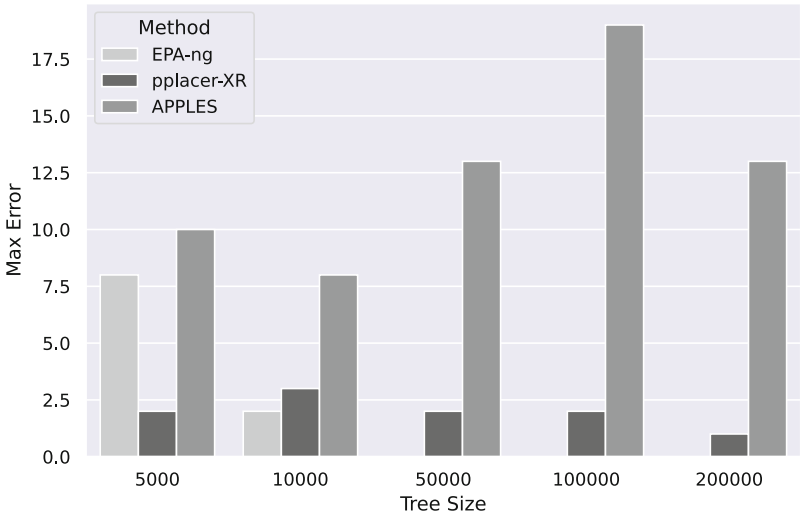

Fig. 3. Maximum delta error over of all placements for each method over five backbone tree sizes. Averages for the 5,000, 10,000, 50,000, and 100,000 taxa trees are over all 1,000 query sequence placements, and for the 200,000 taxa tree the averages are for 200 query sequence placements. EPA-ng was unable to run for trees larger than 10,000 taxa and so no results are reported for EPA-ng on those datasets. This was run on the UIUC campus cluster with 64 GB of memory per placement and 4 h time limit.

error of 3, while EPA-ng and APPLES have a maximum delta error of 8 and 19 respectively.

Finally, Fig. 4 and Table 1 shows the fraction of times that the query is optimally placed (so that delta error is 0); here, too, pplacer-XR outperforms APPLES and EPA-ng, with APPLES having poorer results than EPA-ng.

Table 1. Fraction optimally placed query sequences on backbone trees of size n

	$n = 5000$	$n = 10{,}000$	$n = 50{,}000$	$n = 100{,}000$	$n = 200{,}000$
pplacer-XR	0.858	0.872	0.919	0.918	0.925
EPA-ng	0.781	0.787	X	X	X
APPLES	0.770	0.785	0.846	0.838	0.875

4.2 Time Analysis

One advantage of distance-based APPLES over maximum likelihood placement methods is generally improved runtime [2]. This can be seen in the results from Fig. 5 and Table 4. For the trees with 5,000 taxa, pplacer-XR is able to place

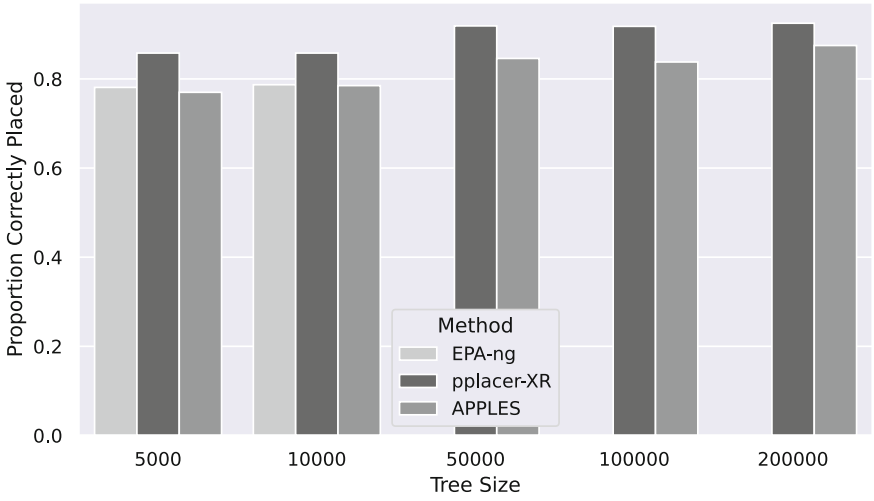

Fig. 4. Fraction of placements with a delta error of zero over of all placements for each method over five backbone tree sizes. Averages for the 5,000, 10,000, 50,000, and 100,000 taxa trees are over all 1,000 query sequence placements, and for the 200,000 taxa tree the averages are for 200 query sequence placements. EPA-ng was unable to run for trees larger than 10,000 taxa and so no results are reported for EPA-ng on those datasets. This was run on the UIUC campus cluster with 64 GB of memory per placement and 4-h time limit.

in half the time of EPA-ng, but takes approximately ten times as much time as APPLES. On trees of 10,000 taxa, the runtime difference between EPA-ng and pplacer-XR grows further to over 3 times the speed, but pplacer-XR is now just 8.3 times slower than APPLES. On the largest tree of 200,000 taxa, pplacer-XR is only 2.7 times slower than APPLES, suggesting the significant speed advantage of APPLES (and potentially other distance-based methods) degrades as the backbone tree size increases. However, on these tree sizes, pplacer-XR was never as fast as APPLES. Finally, EPA-ng was not able to run on the 200,000-taxon backbone tree due to memory limitations and is not shown.

Table 2. Average delta error (Δe) in backbone trees of size n

	$n = 5000$	$n = 10{,}000$	$n = 50{,}000$	$n = 100{,}000$	$n = 200{,}000$
			Δe		
pplacer-XR	0.150	0.132	0.085	0.084	0.075
EPA-ng	0.239	0.219	X	X	X
APPLES	0.366	0.330	0.239	0.247	0.250

Table 3. Maximum delta error (e_{Max}) on backbone trees of size n

	$n = 5000$	$n = 10{,}000$	$n = 50{,}000$	$n = 100{,}000$	$n = 200{,}000$
			e_{Max}		
pplacer-XR	2	3	2	2	1
EPA-ng	8	2	X	X	X
APPLES	10	8	13	19	13

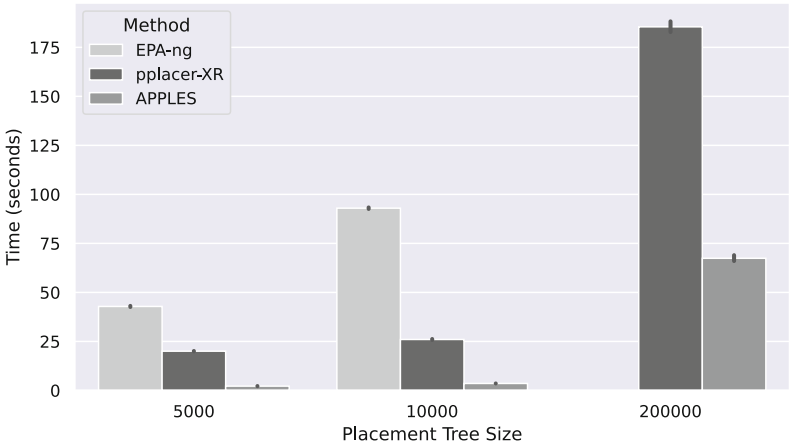

Fig. 5. Average time of all placements for each method over three backbone tree sizes. Averages for the 5,000- and 10,000-taxon trees are over all 1,000 query sequence placements, and for the 200,000-taxon tree the averages are for 200 query sequence placements. EPA-ng was unable to run for trees with 200,000 taxa. This was run on the UIUC campus cluster with 64 GB of memory per placement and 4 h time limit.

Table 4. Average placement time in seconds on backbone trees of size n

	$n = 5000$	$n = 10{,}000$	$n = 200{,}000$
pplacer-XR	20	25	185
EPA-ng	42	92	X
APPLES	2	3	67

5 Conclusions

We have presented pplacer-XR, a technique that uses pplacer, the most accurate placement method, as a step inside a three-stage process for phylogenetic placement. As this study shows, pplacer-XR matches the scalability of APPLES (to our knowledge, the only previous placement method able to perform phylogenetic placement on ultra-large backbone trees with 200,000 leaves), but improves on its accuracy. We also show that pplacer-XR improves on the accuracy, speed,

and scalability of EPA-ng, the previous most scalable likelihood-based placement method. Thus, pplacer-XR is an advance in phylogenetic placement methods for large placement trees.

This study suggests several directions for future work. The main advantage of APPLES over pplacer-XR is speed, where APPLES is definitely better (roughly half to one-third the running time of pplacer-XR). Thus, future work should examine ways to reduce the running time for pplacer-XR. It is also possible that changes to the algorithmic design (e.g., how the placement subtree is defined) could improve the accuracy of pplacer-XR. A major limitation of this study is that we only explored phylogenetic placement when given the true alignment of the query sequences to the backbone sequences and where the query sequences were full length; hence, future work should evaluate the case where the input query sequences are short (perhaps with sequencing errors) and unaligned to the backbone sequences. Another direction for future work is to modify pplacer-XR to handle multiple queries at the same time, a capability that EPA-ng is specifically designed to handle. Finally, we note that APPLES2 [1], an improvement on APPLES, has just been developed, and future work should compare pplacer-XR to APPLES2.

Acknowledgments. The research presented here is the result of a course project by EW and YC for the Spring 2020 course CS 581: Algorithmic Genomic Biology, at the University of Illinois, taught by TW. This work was supported in part by the National Science Foundation grant ABI-1458652 to TW.

Appendix

Commands to create backbone trees: All placement methods use the same backbone tree topologies, but have different branch lengths (following the protocol as provided in [2]). We downloaded the backbone trees with their optimized branch lengths for each phylogenetic placement method from the APPLES repository.

On replicates 1 and 2 of the 100,000-leaf backbone condition and replicate 0 of the 200,000-leaf replicate backbone condition, each containing more than two identical sequences, the backbone trees had polytomies. We randomly resolved these in order to run RAxML.

Random tree refinement: raxmlHPC-PTHREADS -f e -t res_true.fasttree -m GTRGAMMA -s aln_dna.phy -n REF -p 1984 -T 16

APPLES command: run_apples.py -t backbone.tree -s ref.fa -q query.fa -T 16 -o apples.jplace

EPA-ng command: epa-ng –ref-msa ref.fa –tree backbone.tree –query query.fa –outdir $query –model RAxML_info.REF8 –redo -T 16

pplacer-XR commands: python3 pplacer-XR.py GTR RAxML_info.REF backbone.tree output_dir aln.fa query.txt 2000

Github site: https://github.com/chry04/pplacer_plusplus

References

1. Balaban, M., Roush, D., Zhu, Q., Mirarab, S.: APPLES-2: faster and more accurate distance-based phylogenetic placement using divide and conquer. bioRxiv (2021). https://doi.org/10.1101/2021.02.14.431150
2. Balaban, M., Sarmashghi, S., Mirarab, S.: APPLES: scalable distance-based phylogenetic placement with or without alignments. Syst. Biol. **69**(3), 566–578 (2020)
3. Barbera, P., et al.: EPA-NG: massively parallel evolutionary placement of genetic sequences. Syst. Biol. **68**(2), 365–369 (2019)
4. Berger, S.A., Krompass, D., Stamatakis, A.: Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. Syst. Biol. **60**(3), 291–302 (2011)
5. Bik, H.M., Porazinska, D.L., Creer, S., Caporaso, J.G., Knight, R., Thomas, W.K.: Sequencing our way towards understanding global eukaryotic biodiversity. Trends Ecol. Evol. **27**(4), 233–243 (2012)
6. Chaumeil, P.A., Mussig, A.J., Hugenholtz, P., Parks, D.H.: GTDB-Tk: a toolkit to classify genomes with the genome taxonomy database. Bioinformatics **36**(6), 1925–1927 (2020)
7. Conlan, S., Kong, H.H., Segre, J.A.: Species-level analysis of DNA sequence data from the NIH Human Microbiome Project. PLoS ONE **7**(10), e47075 (2012)
8. Matsen, F.A., Kodner, R.B., Armbrust, E.V.: pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinform. **11**(1), 538 (2010)
9. McCoy, C.O., Matsen IV, F.A.: Abundance-weighted phylogenetic diversity measures distinguish microbial community states and are robust to sampling depth. PeerJ **1**, e157 (2013)
10. Mirarab, S., Nguyen, N., Guo, S., Wang, L.S., Kim, J., Warnow, T.: PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences. J. Comput. Biol. **22**(5), 377–386 (2015)
11. Mirarab, S., Nguyen, N., Warnow, T.: SEPP: SATé-enabled phylogenetic placement. In: Biocomputing 2012, pp. 247–258. World Scientific (2012)
12. Nguyen, N.P., Mirarab, S., Liu, B., Pop, M., Warnow, T.: TIPP: taxonomic identification and phylogenetic profiling. Bioinformatics **30**(24), 3548–3555 (2014)
13. Price, M.N., Dehal, P.S., Arkin, A.P.: FastTree 2-approximately maximum-likelihood trees for large alignments. PLoS ONE **5**(3), e9490 (2010)
14. Shah, N., Molloy, E.K., Pop, M., Warnow, T.: TIPP2: metagenomic taxonomic profiling using phylogenetic markers. Bioinformatics (2021)
15. Stamatakis, A.: RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics **22**(21), 2688–2690 (2006)
16. Tavaré, S.: Some probabilistic and statistical problems in the analysis of DNA sequences. Lect. Math. Life Sci. **17**(2), 57–86 (1986)
17. Yang, Z.: Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J. Mol. Evol. **39**(3), 306–314 (1994)