

# SEPP: SATé-Enabled Phylogenetic Placement

S. MIRARAB, N. NGUYEN, AND T. WARNOW\*

*Department of Computer Science  
University of Texas at Austin  
Austin, TX 78712, USA  
\*tandy@cs.utexas.edu  
www.cs.utexas.edu.edu*

We address the problem of Phylogenetic Placement, in which the objective is to insert short molecular sequences (called query sequences) into an existing phylogenetic tree and alignment on full-length sequences for the same gene. Phylogenetic placement has the potential to provide information beyond pure “species identification” (i.e., the association of metagenomic reads to existing species), because it can also give information about the evolutionary relationships between these query sequences and to known species. Approaches for phylogenetic placement have been developed that operate in two steps: first, an alignment is estimated for each query sequence to the alignment of the full-length sequences, and then that alignment is used to find the optimal location in the phylogenetic tree for the query sequence. Recent methods of this type include HMMALIGN+EPA, HMMALIGN+ppplacer, and PaPaRa+EPA. We report on a study evaluating phylogenetic placement methods on biological and simulated data. This study shows that these methods have extremely good accuracy and computational tractability under conditions where the input contains a highly accurate alignment and tree for the full-length sequences, and the set of full-length sequences is sufficiently small and not too evolutionarily diverse; however, we also show that under other conditions accuracy declines and the computational requirements for memory and time exceed acceptable limits. We present SEPP, a general “boosting” technique to improve the accuracy and/or speed of phylogenetic placement techniques. The key algorithmic aspect of this booster is a dataset decomposition technique in SATé, a method that utilizes an iterative divide-and-conquer technique to co-estimate alignments and trees on large molecular sequence datasets. We show that SATé-boosting improves HMMALIGN+ppplacer, placing short sequences more accurately when the set of input sequences has a large evolutionary diameter and produces placements of comparable accuracy in a fraction of the time for easier cases. SEPP software and the datasets used in this study are all available for free at <http://www.cs.utexas.edu/users/phylo/software/sepp/submission>.

*Keywords:* Phylogenetic placement, metagenomic analysis

## 1. Introduction

Metagenomic datasets contain thousands to millions of short sequences, many from different species and for different genes. Determining the species present within a metagenomic dataset and their relative abundances are two of the main objectives in metagenomic analysis. However, these problems do not address other issues, such as the discovery of new species and the inference of evolutionary relationships between the species in the sample. Under the assumption that all short reads are from the same gene and that a tree and alignment for a large number of full-length sequences for that gene are available, each short read can be placed into a phylogenetic tree, thereby enabling species identification for these reads, the inference of evolutionary relationships between the reads, and potentially also the identification of reads coming from unknown species. This is called the “phylogenetic placement” problem, formally stated as follows:

*Phylogenetic Placement Problem.*

- Input: the *backbone* tree  $T$  and alignment  $A$  on set  $S$  of full-length sequences, and query

sequence  $s$ .

- Output: tree  $T'$  containing  $s$  obtained by adding  $s$  as a leaf to  $T$ .

Several methods have been developed for this problem using the following two steps:

- Step 1: insert  $s$  into alignment  $A$  to produce the *extended alignment*  $A'$
- Step 2: add  $s$  into  $T$  using  $A'$ , optimizing some criterion

Methods for the first step include HMMALIGN<sup>1</sup> and the recently introduced PaPaRa<sup>2</sup> method. Methods for the second step include EPA<sup>3</sup> and pplacer,<sup>4</sup> which seek to optimize maximum likelihood (pplacer also provides a Bayesian approach). Methods for phylogenetic placement can therefore be described by how they handle each step. Three such methods include PaPaRa+EPA,<sup>2</sup> HMMALIGN+EPA,<sup>3</sup> and HMMALIGN+pplacer.<sup>4</sup> EPA and pplacer are comparably fast and have almost identical placement accuracy, but have somewhat different memory usage and algorithmic features;<sup>4</sup> hence the differences between HMMALIGN+EPA and HMMALIGN+pplacer do not impact the placement accuracy, and have a minor impact on running time and memory usage. The two techniques for computing the extended alignment, PaPaRa and HMMALIGN, are very different. HMMALIGN computes a HMM to represent the full-length alignment, and then aligns each query sequence to that HMM. In contrast, PaPaRa uses RAxML to estimate ancestral state vectors for each branch in the tree, aligns the query sequence to every ancestral state vector, selects the alignment that had the best score and uses it to extend alignment  $A$  to include  $s$ . Consequently, PaPaRa is computationally more expensive than HMMALIGN,<sup>2</sup> but EPA placements of query sequences based upon PaPaRa extended alignments can be more accurate than EPA placements based upon HMMALIGN extended alignments.

In this study, we introduce *SEPP*, a new basic algorithmic strategy for boosting the performance of methods for Phylogenetic Placement. We use a dataset decomposition within SATé-2<sup>5</sup> to decompose the set of full-length sequences into disjoint sets based upon the input tree and alignment, so that each set contains a small number of closely related sequences. We then compute the extended alignment for each query sequence and the placement of the query sequences into the input tree by running HMMALIGN and pplacer on the subsets instead of on the full tree. Depending upon the decomposition sizes, this approach can result in much faster and more accurate placements, as we will show. The parameters of the divide-and-conquer technique thus allow the user to trade-off running time and accuracy.

We report on a study comparing SEPP to HMMALIGN+pplacer and PaPaRa+pplacer on a collection of simulated datasets (with 500-taxon backbone trees and alignments) and one large biological dataset with a curated alignment of its 13,822 full-length sequences. Our study shows that SEPP is more accurate than PaPaRa+pplacer and HMMALIGN+pplacer when the full-length sequence dataset are evolutionary distant, and yields placements of comparable accuracy but much more efficiently when the sequences are more closely related.

We finish by defining some terms we will use throughout this paper.

- Reference alignment: We use the true alignment for the simulated datasets and the curated alignment for the biological dataset.
- Reference tree: For the simulated datasets, we use the true (model) tree with all zero-event branches collapsed. For the biological dataset, we use the RAxML<sup>6</sup> bootstrap tree computed

on the curated alignment, with all branches having less than 75% bootstrap support contracted. For a given set of full-length sequences and one query sequence, we restrict the reference tree on the full set of sequences to the subset, in order to define the reference topology for that subset.

- Query sequence: the sequences that are not part of the reference alignment, and which will be inserted into the reference tree.
- Backbone tree and alignment: this is the tree and alignment on the set  $S$  of full-length sequences, provided as input (along with the query sequences) to the phylogenetic placement problem.
- Extended alignment: The alignment on  $S \cup \{s\}$  produced by inserting  $s$  into the backbone alignment on  $S$ .

Note: The terms “reference alignment” and “reference tree” are used differently in other papers,<sup>2-4</sup> where these refer to the input alignment and the ML tree estimated on that alignment. Because our study evaluates accuracy with respect to the true tree (known for the simulated data) or to the curated tree (for the biological dataset), we reserve the term “reference tree” for these objects, instead of for trees estimated on estimated alignments.

## 2. SEPP: SATé-Enabled Phylogenetic Placement

SEPP is a meta-method that works with existing methods for the two steps of phylogenetic placement (computing the extended alignment, and placing query sequences into a backbone tree). SEPP utilizes the dataset decomposition technique in SATé-2, a method that co-estimates sequence alignments and phylogenies. This technique takes as input a tree  $T$  and a target size  $K$ , and it partitions the leaf set of  $T$  into smaller subsets, as follows. SATé-2 removes a centroid edge (an edge that splits the taxon set into two approximately equally sized subsets) from the input tree, thus dividing the tree into two subtrees, and repeats the process until each subset has at most  $K$  leaves (taxa). Thus, the taxa within any single subset are close together within the tree and densely sampled within that subset.

The input to SEPP consists of

- the backbone tree  $T$  and alignment  $A$  for the full-length sequences and a set of query sequences,
- positive integers  $a$  and  $p$ , with  $p \geq a$ ,
- a technique for aligning the query sequence to a multiple sequence alignment of full-length sequences, and
- a technique for inserting the query sequence into a backbone tree, given the extended alignment that includes the query sequence.

The output of SEPP is the placement of each query sequence into the backbone tree. In this study we explore SEPP where we use HMMALIGN to produce extended alignments and pplacer to insert query sequences into backbone trees.

We now show how SEPP uses the parameters  $a$  and  $p$  to compute the extended alignment and placement of a set of query sequences into the tree. For the sake of simplicity of exposition, we describe this for a single query sequence.

- We use the SATé-2 dataset decomposition strategy to divide the set of taxa in the tree  $T$  into disjoint subsets of size at most  $p$ . These subsets are called the “taxon-insertion-subsets.”
- We further divide each taxon-insertion subset into smaller subsets of size  $a$ . These subsets are the “alignment-subsets”. Thus, each alignment-subset is a subset of exactly one taxon-insertion-subset.
- We compute the HMM profile using HMMER for each of the alignment-subsets, and we find the alignment-subset that the query sequence  $s$  has the best match to. We use HMMALIGN to produce an alignment of  $s$  to the backbone alignment on the alignment-subset, and then use that alignment to produce the extended alignment for  $S \cup \{s\}$ .
- We find the taxon-insertion-subset that contains the selected alignment-subset, and we use pplacer to locate the query sequence  $s$  into the subtree of the backbone tree induced by the taxon-insertion-subset. Finally, we use the location of  $s$  in the subtree to insert  $s$  into the backbone tree on the entire set of taxa.

Thus, the two parameters  $a$  and  $p$  control the behavior of SEPP. We let  $a$  range from 10 to 250 for the simulated datasets and from 10 to 2500 for the biological dataset. We let  $p$  range from 10 to 500 for the simulated datasets and from 100 to 13,822 for the biological dataset.

### 3. Study Design

We evaluate phylogenetic placement methods on both empirical and simulated datasets. We include HMMALIGN+pplacer and PaPaRa+pplacer as representatives of currently available methods for phylogenetic placement. We also include SEPP used with HMMALIGN to produce extended alignments and pplacer to place the query sequences.

We studied performance of these phylogenetic placement methods on 61 sequence datasets (available at <http://www.cs.utexas.edu/users/phylo/software/sepp/submission>). We included 20 simulated 1000-taxon datasets that have evolved with substitutions and indels from each of three different model conditions (M2, M3, and M4), each with the “medium” gap length distribution (see Liu et al.<sup>7</sup> for these data). The three model conditions are chosen such that one dataset is hard, one is moderate, and one is easy. Because these are simulated datasets, the true alignment and true tree are known for each datasets.

We also used a large bacterial dataset, 16S.B.ALL, with 27,643 16S rRNA sequences, originally taken from the Gutell Comparative Ribosomal Website (CRW),<sup>8</sup> and also studied by Liu et al.<sup>5</sup> This dataset has a curated alignment based upon confirmed secondary (and higher-order) structures, which are highly reliable. We use a ML bootstrap tree as the curated tree for this dataset, retaining only those branches with bootstrap support above 75%.<sup>5</sup> Thus, the 16S.B.ALL dataset has a curated tree and alignment as well.

Each dataset was randomly divided into two subsets of equal size, with one subset ( $S$ ) used to define the backbone alignment and tree, and the other subset ( $R$ ) used to produce the query sequences. These query sequences are created by taking substrings of normally-distributed lengths (from two distributions, described below), and with the start positions chosen uniformly at random.

Two categories of reads are generated for each sequence in the M2, M3, and M4 datasets: “long” reads, with a mean length of 250 and a standard deviation of 60, and “short” reads, with a mean length of 100 and a standard deviation of 20. A total of 10 fragmentary sequences are generated

for each sequence, with half long and half short. Since these datasets each include 500 reference and 500 non-reference sequences, this process yields 2500 short and 2500 long reads per dataset. In summary, each M2, M3, and M4 dataset has a reference tree and alignment with 500 taxa and a total of 5000 fragmentary sequences, of which half are “short” and half are “long”.

For the 16S.B.ALL biological dataset, we create two categories of reads, with length distributions identical to those of simulated datasets. This dataset contains 27,643 taxa, of which we use 13,822 sequences for the backbone tree, leaving us with 13,821 sequences for creating fragmentary reads. For each of these 13,821 sequences, we generated one fragmentary sequence, randomly choosing between the long and short distributions. Thus, for this dataset the backbone tree and alignment has 13,822 taxa, and there are 13,821 fragmentary sequences.

The sequences in  $S$  are used to create two backbone alignments and trees, as follows. For sets  $S$  that are produced by simulating sequence evolution, we have the true alignment and the true tree. We restrict each of these (which have 1000 taxa) to the subset of 500 full-length sequences, and then run RAxML on the resultant tree/alignment pair in order to optimize the branch lengths and GTR+Gamma parameters. This produces the first alignment/tree backbone. The second backbone alignment/tree pair is produced by running SATé on the set of full-length sequences.

For the 16S.B.ALL dataset, we run RAxML on the curated alignment to produce a binary tree. We then restrict the tree to the subset of 13,822 sequences, and optimize the branch lengths and GTR+Gamma parameters on the tree using RAxML. This produces the first backbone alignment/tree pair. We use SATé on the subset of 13,822 full-length sequences to produce the second.

We used SATé to produce these estimated alignment/tree pairs because SATé produces more accurate alignments and trees than any two-phase method (where an alignment is first estimated and then a tree computed on that alignment) for these datasets.<sup>5</sup> We used SATé-2, the new algorithm design for SATé; this produces an alignment and an ML tree on the alignment estimated using RAxML. For the 16S.B.ALL dataset, we used FastTree<sup>9</sup> within SATé-2 in each iteration, and finished with RAxML in order to produce optimized GTR+Gamma parameters on the final tree.

We classify each query sequence for its likely difficulty in phylogenetic placement as follows. We use HMMER to produce a HMM profile for the reference alignment, and then to classify the query sequences with respect to the HMM profile. The fragmentary reads are classified as easy to align (“easy”) if the obtained E-value is less than  $10^{-5}$ , and as “hard” otherwise. Among the hard reads, there are some reads for which HMMER does not report any E-value due to default filtering settings of HMMER. We classify such reads as “very hard” reads. In earlier phylogenetic placement studies, the hard fragments are excluded;<sup>4</sup> however, our study does not automatically eliminate hard fragments. Many very hard reads are able to be placed by SEPP, because the reads will receive E-values with respect to the smaller alignment subsets. Those that fail to be placed at all by SEPP are removed from the experimental study; this process removes 9 from all the simulated datasets together and 5 from the biological dataset. Thus, results for each simulated model condition are for 99997-100000 retained query sequences (20 replicas, each with 5000 query sequences) and 13,819-13,821 retained query sequences for the two 16S.B.ALL datasets.

Table 1 shows various statistics for the true or curated alignment of the datasets included in our study. The p-distance is the fraction of sites within an alignment in which two sequences are different and “% gaps” is the percentage of gaps within the alignment. The empirical statistics show that the

Dataset	Type	Size backbone	Num generated query seqs	Avg p-dist	Max p-dist	% gap
M2	sim	500	5000	0.68	0.76	67
M3	sim	500	5000	0.66	0.74	53
M4	sim	500	5000	0.50	0.60	51
16S.B.ALL	emp	13,822	13,821	0.21	0.52	74

Table 1: Dataset statistics: We present statistics for the true alignments for the simulated datasets (M2, M3, M4) and statistics for the curated alignment on the biological dataset, 16S.B.ALL. However, a small number of query sequences is deleted from some of the runs.

datasets vary substantially in terms of evolutionary distances, with datasets from model M2 having the largest evolutionary distances and 16S.B.ALL having the smallest.

**Measurements.** We measure placement accuracy (averaged over all the query sequences), running time, and peak memory usage, for each method on each dataset. For the simulated datasets we report averages for these measurements over the 20 replicates in each model condition.

*Placement accuracy:* All our phylogenetic placement methods use pplacer to position each query sequence into the backbone tree, and we use the most likely placement computed by pplacer (multiple possible placements for each read, along with the likelihood of each placement, are provided by pplacer.) We then compare the tree that is created to the reference tree, and compute the number of missing branches. This number in isolation is hard to interpret, for at least two reasons. In the case where SATé alignment/tree is the input, the backbone tree itself contains error. The error of the initial backbone tree is a lower bound on the tree error after placement of reads (in fact it is a rather liberal lower bound, as the optimal placement of fragments can still have errors higher than the initial tree). In the case where true or curated alignment/tree is the input, the initial tree has no error, but we can still establish useful lower bounds of the tree error. This can be done by using the reference alignment of query sequences to be the backbone alignment as input to the pplacer. The resulting placement of query sequences is the best we can realistically hope for.

To account for the lower bounds described above, we also define and report the “delta error” for each technique, as follows. For each read  $s$  placed on the SATé backbone tree, we report the difference between the number of missing branches of the initial backbone tree and the number of missing branches after placement of  $s$ . When the backbone tree is the reference (true or curated) tree, we report the difference between the number of missing branches of the tree produced by placement of  $s$  according to the reference alignment of  $s$  to  $S$  and the number of missing branches of the tree after placement of  $s$ . In all cases, the number of missing branches in each tree is defined with respect to the reference tree for the taxa in the given tree. Thus, the number of missing branches in the backbone trees is defined by the reference tree on the set  $S$  of backbone taxa, and the number of missing branches in the tree produced by placing the query sequence into the backbone tree is defined by the reference tree on the set  $S \cup \{s\}$ .

Note that in the case where the backbone tree is the reference tree, the number of missing branches is equal to the node distance between the correct placement of reads and actual placements, the error used in the literature.<sup>2-4</sup> However, this edge distance is not as meaningful as the

number of missing branches with respect to either the true or curated tree, since estimated trees will generally have error.

*Computational aspects:* We report the running times and peak memory usage, each measured separately for the computation of extended alignments and placement of query sequences. These reported values are for placement of all query sequences in each set. Due to memory requirements of PaPaRa and pplacer, 16S.B.ALL experiments are run on a Linux machine with 16 cores and 256GB of main memory. The results for simulated datasets are obtained on a heterogeneous condor cluster.

## 4. Results

*Algorithm design experiments.*

Figure 1 shows results where we vary the two algorithmic parameters  $a$  and  $p$ , with peak memory usage indicated by the size of the circle. Note that decreasing  $a$  to 50 (and sometimes to 10) and increasing  $p$  tends to improve the placement accuracy, but at a running time cost. Also, bigger improvements in accuracy are obtained by decreasing  $a$  than by increasing  $p$ . However, for most conditions, there is a wide range of parameter settings in which the differences in placement error are quite small (often less than half an edge), and within this collection there can be substantial differences in running time.

To set the default parameters, we sought a setting that worked reasonably well with respect to both running time and placement accuracy. Setting  $a = p = 1000$  for the 16S.B.ALL datasets and  $a = p = 50$  for the simulated datasets produced good results. These settings correspond to setting the subset sizes to about 10% of the number of taxa in the backbone tree. Note, however, that setting  $a = p = 50$  is by no means optimal for the M2 model condition (four other settings, have less error and complete faster). Similarly, setting  $a = p = 1000$  is the fastest for the 16S.B.ALL datasets, but more accurate results can be obtained with other settings (with  $a$  below 1000) for a running time cost.

*Comparisons using the default setting for SEPP.*

We present results for PaPaRa+pplacer, HMMALIGN+pplacer, and the default setting for SEPP where we set  $a = p$  to approximately 10% of the number of taxa in the backbone tree. This yields parameters 50/50 for the simulated datasets (backbone trees have 500 taxa) and 1000/1000 for the 16S.B.ALL dataset (backbone trees have 13,822 taxa).

*Results on simulated datasets.* The simulated datasets have backbone trees with 500 sequences and fairly high rates of evolution, with M2 having the highest rate and M4 having the lowest rate (Table 1). Placement error rates were impacted by the model, so that the missing branch rate for all methods is higher on model M2 than on model M3, and higher on model M3 than on model M4 (Table 2). Not surprisingly, absolute error rates are lower with the true alignment and tree than with the SATé alignment and tree. These trends also held for PaPaRa and SEPP.

Figure 2 and Table 2 show results for PaPaRa+pplacer, HMMALIGN+pplacer, and SEPP(50,50) (i.e., SEPP ran with the default setting on this model of  $a = p = 50$ ). Note that SEPP(50,50) has the lowest delta-error of the three methods by far, followed by HMMALIGN+pplacer, and then by PaPaRa+pplacer. Furthermore, the differences are substantial. The methods are clearly also distinguished by running time and peak memory usage. HMMALIGN+pplacer is the fastest, SEPP(50,50) is somewhat slower, and PaPaRa uses much more time. Both PaPaRa+pplacer and

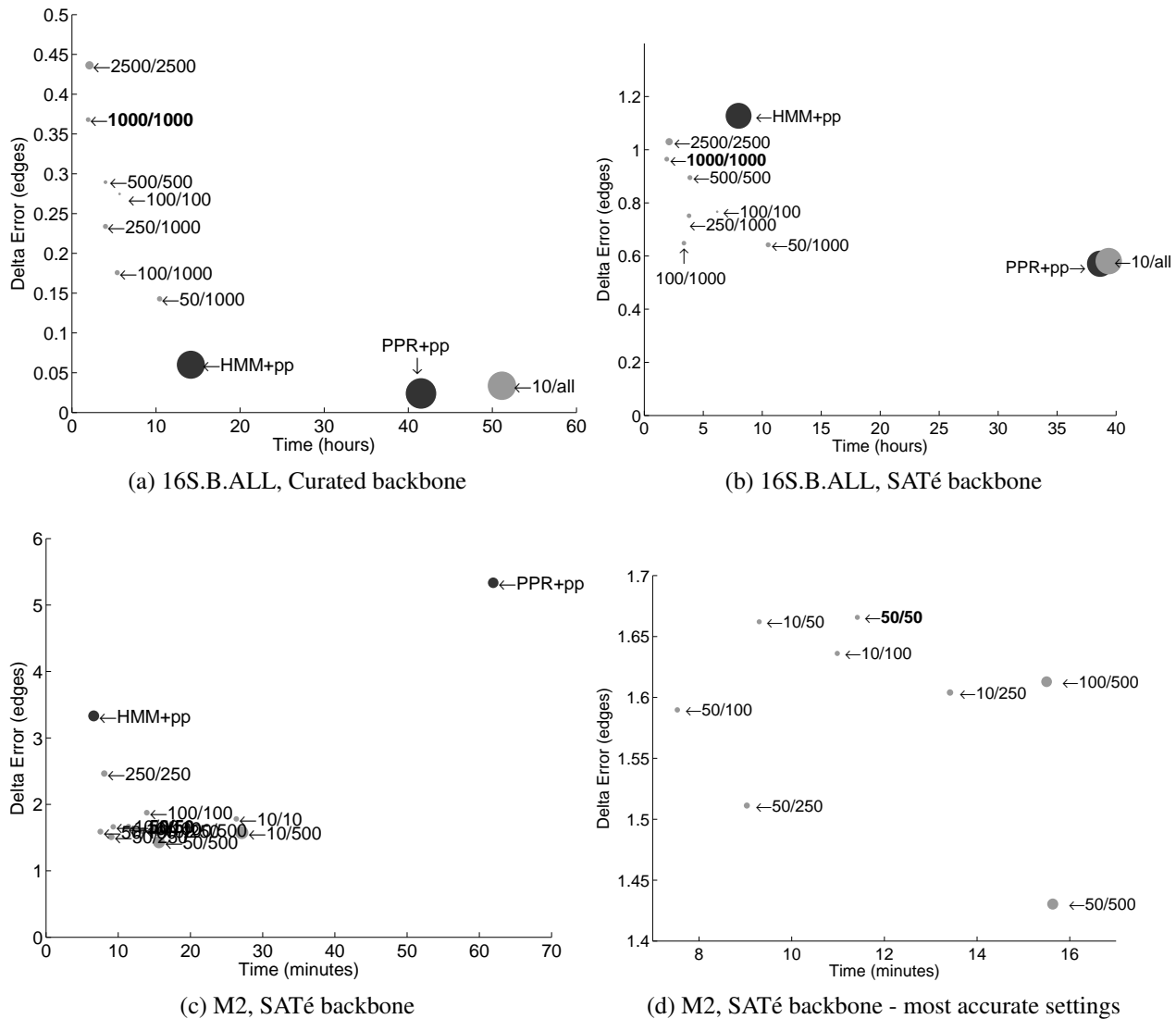


Fig. 1: Scatter plot of delta error (x) versus time (y) versus memory (circle diameters). The symbol “x/y” refers to SEPP(x,y). The default setting is 50/50 for M2 and 1000/1000 for 16S.B.ALL; these points are bold-faced. Note that the default setting for SEPP is far from optimal, with other settings providing better accuracy (and in some cases also better speed).

HMMALIGN+pplacer use more memory than our method.

Results for M3 (see Table 2) are quite similar to M2, with HMMALIGN+pplacer was much more accurate than PaPaRa+pplacer and SEPP(50/50) produced more accurate placements than HMMALIGN+pplacer. However, the gap between SEPP(50,50) and HMMALIGN+pplacer was reduced to only half an edge. On M4 (see Table 2), however, the relative performance between SEPP(50,50) and HMMALIGN+pplacer depended on the backbone tree. For the SATé alignment/tree, SEPP(50,50) was more accurate but slightly slower than HMMALIGN+pplacer. For the true alignment/tree, HMMALIGN+pplacer was somewhat more accurate and took less time. However, the difference in placement accuracy between SEPP(50,50) and HMMALIGN+pplacer was extremely small - less

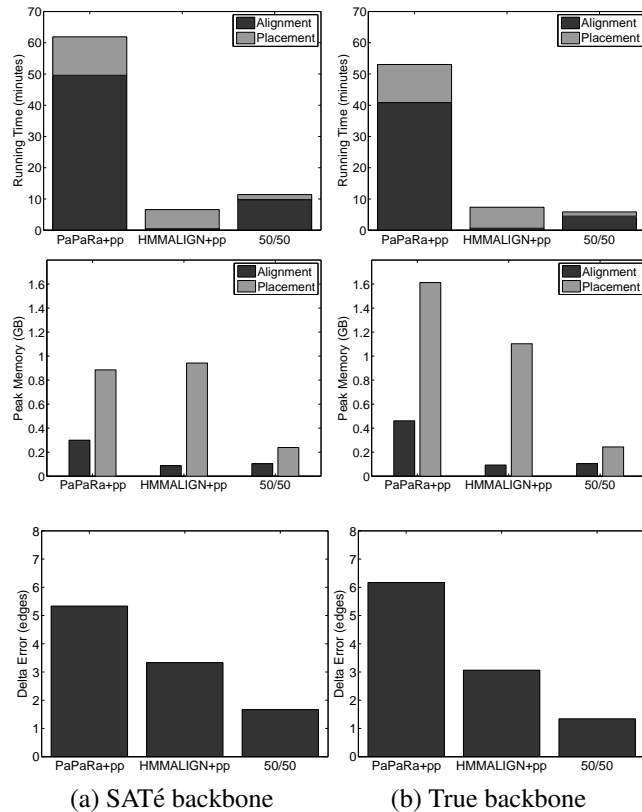


Fig. 2: Results on simulated datasets for model M2. We show running time (top), peak memory usage (middle), and average number of additional missing branches per query sequence (bottom). Results for the SATé backbone alignment and tree are on the left, and results for the true backbone alignment and tree are on the right. The SATé backbone tree has 12.1% missing branch rate and the backbone tree based upon the true alignment has 0.09% missing branch rate. The number of additional missing branches shown (bottom) is the increment above that amount.

than one-ninth of an edge for both backbones.

*16S.B.ALL*. The datasets based upon 16S.B.ALL presented a different kind of challenge. Each dataset had 13,821 query sequences and a backbone tree with 13,822 sequences. Thus, backbone trees were much larger, but the backbone trees and alignments reflected lower rates of evolution.

Figure 3 shows results for both backbones, in which we ran SEPP(1000,1000) (i.e., the default setting). Note that PaPaRa+ppplacer provides a small improvement in placement accuracy (slightly more than half an edge) in comparison to the other methods. However, PaPaRa+ppplacer is enormously computationally intensive, using 40 hours to analyze these data, much longer than either other method. Also, HMMALIGN+ppplacer and PaPaRa+ppplacer have very large peak memory usage, near 60GB on both backbone trees. Thus, PaPaRa+ppplacer is computationally extremely intensive, but both HMMALIGN+ppplacer and PaPaRa+ppplacer have very large peak memory usage.

A comparison of SEPP(1000,1000) to HMMALIGN+ppplacer shows that both have extremely good placement accuracy, with delta-error approximately one edge for both methods on the SATé backbone tree and well under half an edge on the curated backbone tree. HMMALIGN+ppplacer pro-

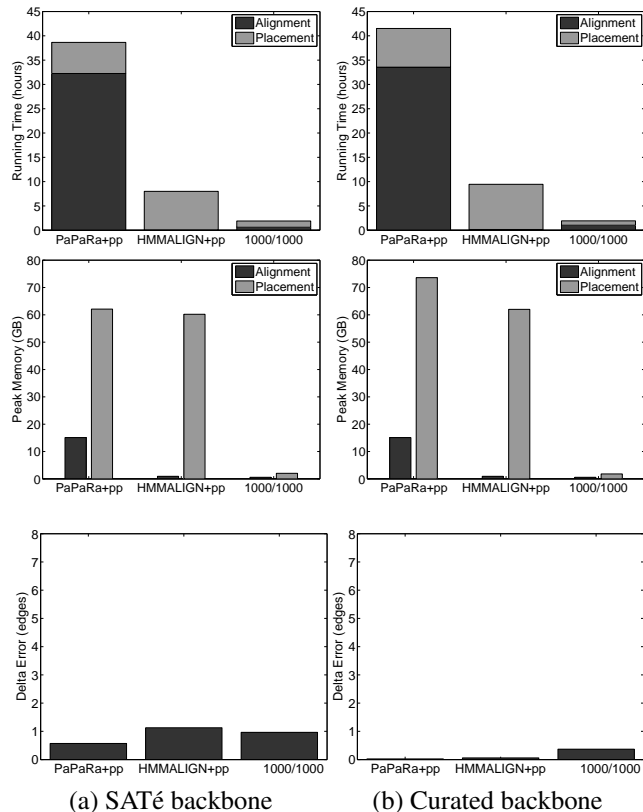


Fig. 3: Results on 16S.B.ALL. We show running time (top), peak memory usage (middle), and average number of additional missing branches per query sequence (bottom). Results for the SATé backbone alignment and tree are on the left, and results for the curated backbone alignment and tree are on the right. The SATé missing branch rate is 7.64%, and the missing branch rate for the backbone tree defined by the true alignment is 1.835%. The number of additional missing branches shown (bottom) is the increment above that amount.

duces more accurate placements than our method for the curated backbone and SEPP(1000,1000) produces more accurate placements for the SATé backbone, but the differences between the two methods are small in both cases (less than a third of an edge). The methods are, however, distinguished by their computational requirements, as HMMALIGN+pplacer is much slower (at least 4 times as much time) and uses dramatically more memory (60GB as compared to about 2GB).

#### Comparing methods on query sequences of different levels of difficulty.

Table 2 compares methods in terms of their placement accuracy as a function of the level of difficulty in placing the query sequence, as predicted by HMMER (see the discussion in Section 3). Note that error increases as the reads become more difficult, as HMMER predicts. We see that SEPP run in default mode performs very well in general (as observed earlier) in comparison to HMMALIGN+pplacer and PaPaRa+pplacer, but has a particularly strong advantage on the hard and very hard reads. Interestingly, PaPaRa+pplacer does well on hard and very hard reads for 16S.B.ALL but not on simulated datasets.

	All reads				Hard reads				Very hard reads			
	bio.	M2	M3	M4	bio.	M2	M3	M4	bio.	M2	M3	M4
	SATé Backbone											
count	13819	99998	99999	99999	104	79510	58924	3989	21	63613	40495	844
HMM	1.1	3.4	1.4	<b>0.3</b>	2.4	4.2	2.3	<b>0.7</b>	3.9	5.2	3.2	1.5
PPR	<b>0.6</b>	5.4	3.6	0.4	<b>0.8</b>	5.8	4.4	1.0	<b>0.9</b>	6.2	4.9	1.5
SEPP	1.0	<b>1.7</b>	<b>1.0</b>	0.4	2.4	<b>2.0</b>	<b>1.4</b>	0.8	3.8	<b>2.4</b>	<b>1.8</b>	<b>1.0</b>
	True or Curated Backbone											
count	13818	99997	99999	99999	104	79511	58924	3989	21	63614	40495	844
HMM	<b>0.0</b>	3.2	1.2	<b>0.0</b>	0.5	4.0	2.0	<b>0.2</b>	2.2	5.0	2.9	0.9
PPR	<b>0.0</b>	6.2	3.8	0.2	<b>0.1</b>	6.7	4.7	0.5	<b>0.0</b>	7.1	5.2	0.9
SEPP	0.4	<b>1.4</b>	<b>0.7</b>	0.1	1.1	<b>1.7</b>	<b>1.1</b>	0.4	1.5	<b>2.0</b>	<b>1.4</b>	<b>0.5</b>

Table 2: Mean delta-error for different categories of query sequences. We show the mean delta-error for each method on each model condition, as a function of the level of difficulty for the query sequence, as estimated by HMMER. Count refers to the number of query sequences processed and placed, HMM refers to HMMALIGN+pplacer, PPR refers to PaPaRa+pplacer, and SEPP refers to SEPP run in default mode.

## 5. Discussion

There are several observations we can make. First, the methods we evaluated for phylogenetic placement—PaPaRa+pplacer, HMMALIGN+pplacer, and SEPP methods—often produce placements that are extremely accurate, increasing the topological error over the input backbone tree by at most an edge (often much less than an edge) on average. Furthermore, while these methods do sometimes have differences in placement accuracy that go beyond an edge, these differences are sometimes still small enough to be relatively unimportant, compared to the computational cost required to obtain the improved placement accuracy.

However, we did observe conditions in which the differences in placement accuracy were quite large, suggesting that increased effort in placing query sequences correctly was merited. For example, we see big differences in placement accuracy on model M2, resulting in several edges improvement produced by SEPP(50,50) over HMMALIGN+pplacer. The conditions under which accuracy differences are substantial are characterized by large evolutionary distances between some pairs of full-length sequences. We conjecture that in such conditions, the HMMs produced by HMMER on the full set of taxa may not be sufficient to produce highly accurate alignments for the query sequences, and will result in degraded placement accuracy. The technique we introduce here avoids this problem by using HMMER to produce HMMs only on smaller, less diverse, subsets of the taxa. As a result, the HMMs may produce more accurate alignments to the query sequences, and result in improved phylogenetic placement.

We note also the interesting differences between HMMALIGN+pplacer and PaPaRa+pplacer. The only dataset on which PaPaRa+pplacer produces more accurate placements than HMMALIGN+pplacer is 16S.B.ALL, while PaPaRa+pplacer has substantially less accurate placements for the faster evolving datasets. We conjecture that the reason for this difference is that 16S.B.ALL has a low evolutionary diameter, while the simulated datasets have larger evolutionary diameters, and the estimation of transition state matrices on each edge may only be highly accurate when the backbone

tree has a low evolutionary diameter.

Furthermore, these methods differ dramatically with respect to running time, with PaPaRa+pplacer much more computationally intensive than HMMALIGN+pplacer and default SEPP, thus suggesting that PaPaRa+pplacer is unlikely to be useful in largescale metagenomic analyses.

The comparison between HMMALIGN+pplacer and SEPP is more complex, because SEPP is parameterized by the two algorithmic parameters  $a$  and  $p$ . Here we see that some very simple settings for these parameters ( $a = p$ , both set to about 10% of the number of taxa in the backbone tree) produces very fast results with generally very good accuracy, coming close to the accuracy obtained by the best methods (or improving on them), but in a fraction of the time. Other settings for the parameters can improve the placement accuracy but require greater running time and memory usage.

It is also helpful to put the running time results in context. New sequencing technologies have enabled gathering millions of short reads from microbial communities, and a question of great practical importance is whether analyzing a million reads is feasible in a reasonable time frame. Since the placement problem treats reads independently, the running time of tools developed for the placement problem increases linearly with the number of reads (in fact most tools have pre-processing steps, the cost of which is constant with regards to the number of reads). Therefore by a simple interpolation one can estimate how long it would take to place a million reads given our reported results for 13,000 reads. If we were to place 1 million reads on a tree of about 13,000 reference sequences, PaPaRa+pplacer would take about 133 days and HMMALIGN+pplacer about 30 days. SEPP on the other hand would take just about 6 days, which should be feasible for many applications. SEPP therefore can enable the analysis of very large data sets in relatively short time frames.

## 6. Conclusions and Future Work

The method we have presented, SEPP, is a general technique for boosting the accuracy and/or speed of a phylogenetic placement method. In this study, we explored its performance when coupled with HMMALIGN and pplacer, and showed improvements in placement accuracy and/or running time. Given the plans to analyze millions of reads, the speed-ups that SEPP provides could be essential to providing scalability for phylogenetic placement methods. Improved accuracy, furthermore, might be obtained by coupling SEPP with PaPaRa for those cases where the backbone tree and alignment has a small enough evolutionary diameter to produce highly accurate extended alignments.

## References

1. S. R. Eddy, *Bioinformatics* **14**, 755 (1998).
2. S. A. Berger and A. Stamatakis, *Bioinformatics* (2011).
3. S. A. Berger, D. Krompass and A. Stamatakis, *Systematic Biology* **60**, 291 (May 2011).
4. F. Matsen, R. Kodner and E. V. Armbrust, *BMC Bioinformatics* **11**, 538+ (October 2010).
5. K. Liu, T. Warnow, M. Holder, S. Nelesen, J. Yu, A. Stamatakis and C. Linder, *Systematic Biology* (2011), in press.
6. A. Stamatakis, *Bioinformatics* **22**, 2688 (2006).
7. K. Liu, S. Raghavan, S. Nelesen, C. R. Linder and T. Warnow, *Science* **324**, 1561 (June 2009).
8. J. Cannone, S. Subramanian, M. Schnare, J. Collett, L. D'Souza, Y. Du, B. Feng, N. Lin, L. Madabusi, K. Muller, N. Pande, Z. Shang, N. Yu and R. Gutell, *BMC Bioinformatics* **3**, p. 2 (2002).
9. M. N. Price, P. S. Dehal and A. P. Arkin, *PLoS ONE* **5**, p. e9490 (03 2010).