eMAFFTadd: scaling MAFFT-linsi-add to large datasets

Chengze Shen, Baqiao Liu, and Tandy Warnow

Abstract

Multiple sequence alignment is essential for many biological downstream analyses. Yet, accurate alignments on large datasets are challenging and can require very long running times; since new sequence data are frequently and regularly obtained, this calls for methods that can add sequences into large alignments rather than requiring the re-estimation from scratch. In addition, sequence datasets exhibiting substantial sequence length heterogeneity are also difficult to align with high accuracy. Methods, such as UPP, have been able to provide good accuracy and operate by extracting a subset of the sequences deemed to be full length, aligning that subset (thus producing a "backbone alignment"), and then adding the remaining sequences into the backbone alignment. There are also standalone methods, such as MAFFT–add, that can add sequences into backbone alignments, but the best version of this method (which uses –linsi) is computationally intensive. Because adding sequences into alignments is a basic and important step in bioinformatics analyses, the development of new approaches with high scalability and accuracy is important. In this study, we present a new sequence-adding method, eMAFFTadd, that achieves high accuracy and scalability. In essence, eMAFFTadd is a way of scaling MAFFT-linsi-add to large datasets. We show that eMAFFTadd is more accurate than UPP, can run on datasets too large for MAFFT-linsi-add, and is fast enough to use on very large sequence datasets. Our software for eMAFFTadd is available in open source shape at `https://github.com/c5shen/eMAFFTadd`.

1 Introduction

Multiple sequence alignment (MSA) is a crucial precursor to many downstream biological analyses, such as phylogeny estimation [13], RNA structure prediction [16], protein structure prediction [5], etc. Obtaining an accurate MSA can be challenging, especially when the dataset is large (i.e., more than 1000 sequences), has a high rate of evolution, or when there is sequence length heterogeneity. Some of these issues have been reasonably well addressed through the newer alignment methods, but sequence length heterogeneity (whether resulting from large indels in the evolution of the sequences or through the inclusion of reads or incompletely assembled gene sequences) still presents significant challenges for accuracy [14, 18, 20]. Finally, many evolutionary biologists are interested in adding new sequences into already estimated alignments and trees rather than recomputing these from scratch. Thus, two MSA problems remain inadequately solved: Problem 1: aligning datasets that contain substantial sequence length heterogeneity, and Problem 2: adding new sequences into an existing alignment.

UPP [14] is an MSA method that was designed to address Problem 1 (i.e., aligning sequence datasets with substantial sequence length heterogeneity). UPP operates in two stages: first, it selects and aligns a small subset of the sequences it considers "full-length" (and hence easy to align well using standard methods), thus producing a "backbone alignment." It then builds an "ensemble of Hidden Markov Models" (eHMM) for the backbone alignment and uses that ensemble to align the remaining sequences. This two-stage design enables UPP to be more accurate than standard alignment methods, such as MAFFT-linsi [8] and PASTA [11], when aligning datasets with fragmentary sequences [14, 17]. Note, therefore, that UPP addresses Problem 1 by using a particular method (which we can refer to as UPP-add) for Problem 2.

UPP-add is not the only method for Problem 2. In addition to its predecessors (the alignment step in SEPP [10]), the most well known techniques for Problem 2 can be found in the MAFFT package [7]. Among the MAFFT --add variants, MAFFT-linsi --add is perhaps the most accurate but also very computationally expensive and is not considered usable beyond "a few hundreds of sequences" [6].

This study aims to design new methods for adding sequences into a backbone alignment to advance methods for both problems. Our approach is a divide-and-conquer technique for making MAFFT-linsi-add scalable so that it can be used on large backbone alignments containing several thousands of sequences. We show, through an experimental study involving large biological and simulated datasets, that this method – eMAFFTadd – provides superior accuracy to both UPP and MAFFT-insi-add, scales to large datasets, and is reasonably efficient (slower than UPP but still fast enough to be used on very large datasets).

The rest of the study is structured as follows. In Section 2, we briefly describe and discuss existing sequence-adding methods. In Section 3 we introduce our new method, eMAFFTadd. In Section 4, we describe the experimental study, and in Section 5 we present the results. We discuss trends in Section 6 and conclude with directions for future work in Section 7.

2 Background

2.1 Overview

The input to Problem 2 is an alignment M on m sequences and a set of q query sequences to add into the alignment. The output is a multiple sequence alignment on all $m + q$ sequences, while the backbone alignment is preserved. In this section, we describe and discuss backgrounds of three popular sequence-adding methods, namely UPP-add, MAFFT-add, and MAFFT-linsi-add.

2.2 Existing sequence-adding methods

2.2.1 UPP-add

UPP-add is essentially the second stage of UPP [14]. As briefly described in Section 1, UPP-add creates a set of HMMs (i.e., eHMM) to add query sequences. It first computes a backbone tree with FastTree2 [15] using the backbone alignment. Then, it recursively decomposes the backbone tree to smaller sub-trees until the last decomposition results in sub-trees with no more than A leaves ($A = 10$ in default UPP). An HMM is created using *hmmbuild* from the HMMER suites [3] for each sub-alignment defined by leaves of each sub-tree. This results in a collection of HMMs. Each query sequence is searched against all HMMs using *hmmsearch*, mapped to the HMM with the highest bit-score (i.e., the most relevant sub-alignment), and aligned using *hmmalign*. Finally, all query alignments are merged transitively to the backbone alignment to form the final alignment. UPP-add needs to compute a FastTree tree on the backbone alignment, which scales sub-quadratically [15] with the number of sequences in the backbone. Then, it scales linearly with the number of query sequences.

2.2.2 MAFFT-add

MAFFT-add in its default setting uses a standard progressive alignment procedure with two iterations to add query sequences. In each iteration, it computes an $(m+q) \times (m+q)$ distance matrix on sequences from both the backbone and queries using shared 6-mers. Then, it computes a guide tree using the distance matrix and builds an alignment. More specifically, for each node in the guide tree, MAFFT-add does an alignment computation only if a query sequence is involved at the node (i.e., at least one child has some query sequences). Otherwise, it simply uses the alignment from the backbone.

There are other variants of MAFFT-add that use more accurate distance calculation and can result in improved alignment accuracy (at the cost of scalability). One of the most accurate variants is MAFFT-linsi-add which we briefly describe below.

2.2.3 MAFFT-linsi-add

The procedure to add sequences using MAFFT-linsi-add is similar to MAFFT-add, with two differences. Firstly, MAFFT-linsi-add uses *localpair* (local pairwise alignment scores) for the distance matrix calculation, which is more accurate than shared 6-mers. Secondly, it only runs for one iteration of progressive

alignment and uses at most 1000 iterations of iterative refinement after the progressive alignment finishes.

Both MAFFT-add and MAFFT-linsi-add theoretically scale quadratically due to the $O((m+q)^2)$ distance matrix calculation. MAFFT-linsi-add is even less scalable since its distance calculation is accurate but costly. Also, it does many steps of refinement that further impact the runtime.

3 Our new method: eMAFFTadd

Since MAFFT-linsi-add is accurate but very computationally expensive, we develop a new method, named eMAFFTadd, to make it scalable even on datasets with several thousands of sequences. The core idea of eMAFFTadd is to break the sequence-adding problem into multiple smaller sub-problems, on which MAFFT-linsi-add can efficiently and accurately run (i.e., divide-and-conquer). Given the recommendation for MAFFT-linsi-add on the MAFFT webpage [6], we limit each sub-problem to at most 500 sequences. Additionally, we arbitrarily set a lower bound of sub-alignment sizes to 50 and have a parameter, u, that can be set by the user as the upper bound on the sub-alignment size. Based on the motivations above, we design the eMAFFTadd pipeline as follows (with one free parameter, u):

1. Build an eHMM as done by UPP-add, but instead of $A = 10$, use $A = 50$ as the decomposition stopping criterion.

2. Instead of using all HMMs/sub-alignments, use sub-alignments of sizes at most u (note that the result is that all sub-alignments have size between 50 and u).

3. Assign query sequences to the selected HMMs/sub-alignments the same way as UPP-add (i.e., best fitting HMM for each query based on bit-scores).

4. Use MAFFT-linsi-add to add assigned query sequences to each sub-alignment. If the number of sequences in the sub-alignment plus the number of assigned queries exceeds 500, then evenly divide the assigned queries into smaller subsets so that each sub-problem has at most 500 sequences.

Note that this design achieves two properties. All sub-problems on which we run MAFFT-linsi-add have at most 500 sequences in total (considering both the backbone sub-alignment and the query sequences), but also, the query sequences are assigned to a sub-problem for which they are likely to be closely related (based on the bit score calculation). These two properties together make for sub-problems that are small enough for MAFFT-linsi-add to run well on, but closely related enough to potentially improve accuracy.

4 Experimental design

4.1 Overview

We conducted four experiments. In the first experiment, we showed the scalability issue with MAFFT-linsi-add when adding many sequences to an existing

alignment. Experiments 2 and 3 address Problem 1, which is the alignment of datasets with sequences of heterogeneous lengths, and Experiment 4 addresses Problem 2, which is adding sequences into a given alignment (not necessarily in the context of aligning datasets exhibiting substantial sequence length heterogeneity). Experiment 2 is used for the design phase of eMAFFTadd, where we tune the parameter u for eMAFFTadd (i.e., the upper bound for sizes of sub-alignments). Experiment 3 compares the pipeline (similar to UPP) based on eMAFFTadd to other MSA methods for aligning datasets with sequence length heterogeneity. Finally, in Experiment 4, we compare eMAFFTadd to other sequence-adding methods on biological datasets in a *leave-many-out* analysis.

All analyses were run on the UIUC Campus Cluster, with 16 cores and 64 GB of memory, and the runtime limit is 12 hours. See Appendix Section A for exact commands of all methods.

4.2 Datasets

We use simulated and biological nucleotide datasets to evaluate eMAFFTadd, while both exhibit sequence length heterogeneity. Empirical statistics for these datasets can be found in Table 1, where a higher p-distance (i.e., normalized Hamming distance) means higher rates of insertions, deletions, and substitutions (i.e., higher rates of evolution).

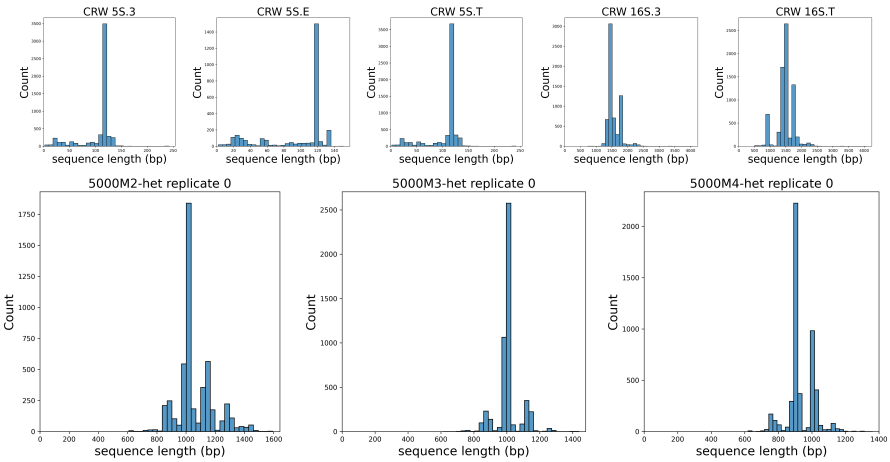

Figure 1: Sequence length histograms for CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets (biological datasets, top), and 5000M2, 5000M3, and 5000M4 dataset on their first replicates (simulated datasets, bottom).

We generated three new simulated conditions with 5000 sequences with a type of evolutionary sequence length heterogeneity, varying the rate of evolution and hence alignment difficulty. We name the conditions 5000M2-het, 5000M3-het, and 5000M4-het, ordered from having the highest rate of evolution (hardest) to the lowest (easiest), with the "het" suffix signaling heterogeneity in the sequence lengths, a common trend in biological datasets (see Figure 1 top). Specifically, our model for sequence length heterogeneity assumes that insertion/deletion ("indel") events, with a small probability, can be promoted to long indel events modelling infrequent large gain or loss of genic regions during the

Dataset	# of Seqs	p-dist.		% gaps	Seq. length	Alignment length
		Avg.	Max			
Simulated nucleotide datasets						
5000M2-het(10)*	5000	0.686	0.789	97.9	1025	48,887
5000M3-het(10)	5000	0.664	0.764	96.0	1001	25,262
5000M4-het(10)	5000	0.528	0.671	96.0	1001	24,415
Biological nucleotide datasets (CRW)						
5S.3(1)	5507	0.418	1.000	74.5	106	414
5S.E(1)	2774	0.305	1.000	87.8	96	793
5S.T(1)	5751	0.425	1.000	75.6	106	436
16S.3(1)	6323	0.315	0.833	82.1	1557	8716
16S.T(1)	7350	0.345	0.901	87.4	1492	11,856

Table 1: Dataset empirical statistics of simulated and biological datasets. CRW is short for the comparative ribosomal website [1]. Datasets marked with (*) denote training datasets. p-dist.: p-distance (i.e., normalized Hamming distance).

evolutionary process (e.g., domain level indels). We then simulated sequences evolving under non-ultrametric model trees using INDELible [4]. The details of the generation can be found in the Appendix (Section B), and their sequence length distributions are presented in Figure 1 (bottom). We use 5000M2-het for the demonstration of the scalability issue with MAFFT-linsi-add (Experiment 1) and parameter training for eMAFFTadd (Experiment 2), and 5000M3-het and 5000M4-het for testing (Experiment 3).

We use five biological datasets, namely 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T, from the comparative ribosomal website (CRW) [1]. These datasets have been used in previous studies [2, 14, 18], exhibit sequence length heterogeneity (Figure 1, top), and have reference alignments based on secondary structures. We use these CRW datasets for testing (Experiments 3 and 4).

4.3 Methods for comparison

We compare eMAFFTadd to existing sequence-adding methods that are introduced in Section 2, including UPP-add [14], MAFFT-add, and MAFFT-linsi-add [7], in terms of alignment accuracy of added query sequences to the same backbone alignment.

4.4 Backbone alignment and query sequences

Experiments 1 to 3 address Problem 1, which is the alignment of datasets with sequences of heterogeneous lengths. We follow the UPP procedure to split each sequence dataset into a backbone set and a query set. For each dataset, the backbone set contains at most 1000 full-length sequences (i.e., lengths 25% around the median length), while the query set contains all remaining sequences (e.g., several thousands of sequences). We compute an alignment on the backbone set using MAGUS [19], because it is the current state-of-the-art alignment method that is accurate and robust to sequence length heterogeneity [18]. The alignment itself is not restricted to MAGUS and can be done by any other accurate alignment methods, such as MAFFT-linsi [8] and PASTA [11].

Experiment 4 addresses Problem 2, which is the problem of adding sequences into an existing alignment (not necessarily in the context addressed by UPP). In this analysis, we use five biological datasets from the CRW [1] collection, each with a reference alignment. For each reference alignment, we leave 100 sequences out as query sequences and use the reference alignment on the remaining sequences as the backbone alignment. For each CRW dataset, we consider three types of queries: 1) when queries are 100 randomly selected sequences, 2) when queries are the shortest 100 sequences, and 3) when queries are the longest 100 sequences.

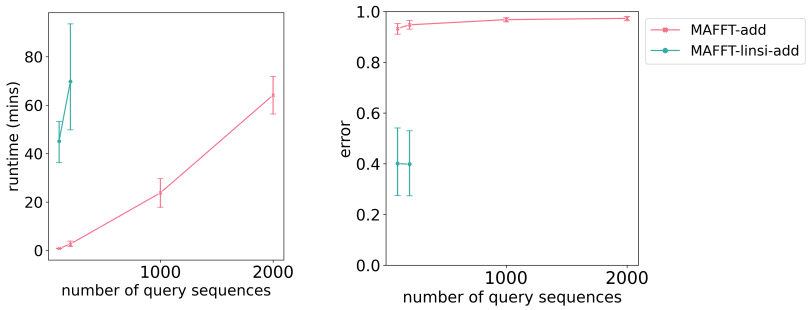

Figure 2: Experiment 1: Runtime (left) and alignment error (right) vs. number of query sequences for MAFFT-add and MAFFT-linsi-add. Both methods add q query sequences to the same 1000-taxon backbone alignment of 5000M2-het (10 replicates). The data points from left to right for each line indicate the addition of 100, 200, 1000, or 2000 query sequences correspondingly. We exclude replicate 4 because MAFFT-linsi-add encountered out-of-memory issues when adding 100 or 200 query sequences. Additionally, MAFFT-linsi-add either encountered out-of-memory issues or did not complete within 12 hours when adding 1000 or 2000 query sequences and thus is not shown.

4.5 Evaluation metrics

We evaluate all methods' accuracy by alignment error, SPFN, and SPFP on added query sequences. A pair of homologies refers to an aligned column between two sequences. The sum-of-pairs false-negative (SPFN) rate is the fraction of pairs of homologies in the reference alignment but missing in the estimated alignment. The sum-of-pairs false-positive (SPFP) rate is the fraction of pairs of homologies that are in the estimated alignment but missing in the reference alignment. Alignment error is the average of SPFN and SPFP. These error rates are calculated using FastSP [12]. We also report wall-clock running time in minutes of all methods.

5 Results

5.1 Experiment 1 - scalability of MAFFT-linsi-add

In this experiment, we explore what sizes of datasets MAFFT-linsi-add can align without crashing or encountering memory issues. We use our training dataset 5000M2-het as the benchmark and either MAFFT-add or MAFFT-linsi-add to add 100, 200, 1000, or 2000 query sequences to the backbone.

Figure 2 shows the results for runtime and alignment error vs. numbers of queries on 5000M2-het datasets. Replicate 4 is excluded because MAFFT-linsi-add encountered out-of-memory issues when adding 100 or 200 query sequences, while MAFFT-linsi-add also encountered out-of-memory issues or did not complete within 12 hours when adding 1000 or 2000 query sequences and thus is not shown.

The results justify the effort to make MAFFT-linsi-add scalable, since MAFFT-linsi-add is more accurate than MAFFT-add but cannot run when there are many query sequences.

5.2 Experiment 2 - tuning parameter

In this experiment, we explore different values for the parameter u, which controls the upper size limit of sub-alignments we use for MAFFT-linsi-add. We vary u between 100, 200, and 400, on our training dataset 5000M2-het (10 replicates).

Appendix Figure 7 shows SPFN, SPFP, and alignment error of eMAFFTadd with different settings for u on 5000M2-het (10 replicates). The differences between eMAFFTadd variants are negligible on all three evaluation metrics. The runtime comparison is also presented in Appendix (Figure 8). We observe that larger values for u generally increases running time. Since larger u (i.e., $u \in \{200, 400\}$) also has outliers with much longer running time without substantially benefiting alignment accuracy, we choose $u = 100$ for eMAFFTadd and use it in the following experiments.

5.3 Experiment 3 - addressing Problem 1

In this experiment, we compare eMAFFTadd (with $u = 100$) to UPP-add [14], MAFFT-add, and MAFFT-linsi-add [7], in the context of *de novo* alignments with sequence length heterogeneity. For each dataset, we follow the UPP procedure to extract at most 1000 full-length sequences to form the backbone alignment (aligned with MAGUS [19]) and add the remaining sequences using each method. We ensure all methods use the same 1000-taxon backbone alignment and measure their alignment accuracy on the added queries. We first show alignment results on the two INDELible simulated datasets (5000M3-het and 5000M4-het), and then on the five CRW datasets (5S.3, 5S.E, 5S.T, 16S.3, and 16S.T). For MAFFT-linsi-add, we only show it for those datasets on which it completes on all replicates.

5.3.1 Alignment results on simulated datasets

Figure 3 shows alignment errors of the methods mentioned above on 5000M3-het and 5000M4-het, each with 10 replicates. MAFFT-linsi-add is not shown

because it either encountered out-of-memory issues or failed to complete within the time limit. On 5000M3-het, MAFFT-add has an error close to 1. Both eMAFFTadd and UPP-add have low errors, and eMAFFTadd is the most accurate. On 5000M4-het, absolute errors for all methods decrease noticeably compared to 5000M3-het, and MAFFT-add has an error similar to eMAFFTadd and UPP-add. eMAFFTadd is still the most accurate, but gaps between methods are much smaller.

Figure 4 provides a closer look at SPFN and SPFP of the best two methods on 5000M3-het and the best three methods on 5000M4-het. On 5000M3-het, eMAFFTadd is more accurate than UPP-add in terms of both SPFN and SPFP, and its advantage in SPFN is quite noticeable. On 5000M4-het, eMAFFTadd and UPP-add both have the lowest SPFP, while eMAFFTadd still has a noticeable advantage over the other two methods in terms of SPFN.

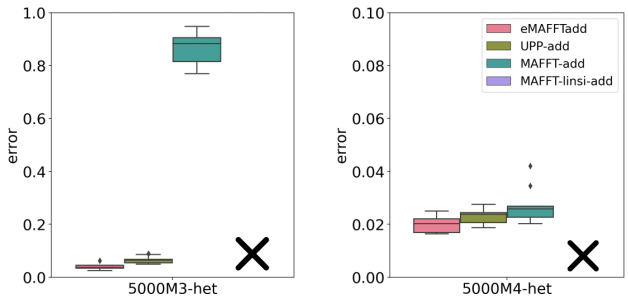

Figure 3: Experiment 3: Alignment error (average of SPFN and SPFP) of different sequence-adding methods on query sequences of 5000M3-het (left) and 5000M4-het (right), each with 10 replicates. Results for MAFFT-linsi-add are not shown because it either failed to complete within 12 hours or encountered out-of-memory issues (marked as "X").

5.3.2 Aligning biological datasets

Figure 5 shows alignment errors for the same set of methods on the five CRW datasets (5S.3, 5S.E, 5S.T, 16S.3, and 16S.T). MAFFT-linsi-add encountered out-of-memory issues on 16S.3 and 16S.T, and is only shown for 5S.3, 5S.E, and 5S.T. The trend is similar to Figure 3, as eMAFFTadd is the most accurate on all five datasets. MAFFT-linsi-add, when it can complete, is on par with eMAFFTadd.

The trends for SPFN and SPFP are also similar to what are presented in the simulated datasets. eMAFFTadd has the lowest SPFN, and MAFFT-linsi-add ties with it if completed within the time limit. On the other hand, SPFP of eMAFFTadd is either on par or slightly higher than UPP-add but always better than MAFFT-add (Appendix Figures 9 and 10).

5.3.3 Runtime comparisons

A runtime comparison in Experiment 3 is presented in Figure 6 and Appendix Figure 11 for simulated datasets and biological datasets. MAFFT-add is the

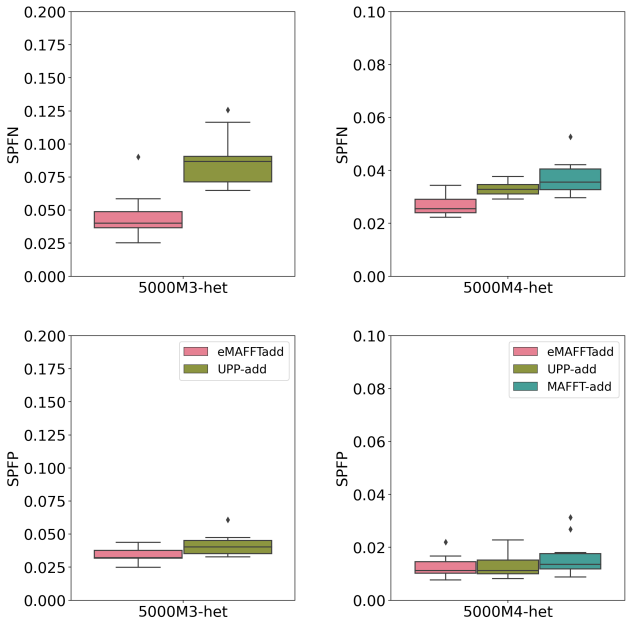

Figure 4: Experiment 3: SPFN (top) and SPFP (bottom) of the best methods on query sequences of 5000M3-het and 5000M4-het, each with 10 replicates. MAFFT-add is not shown for 5000M3-het because it has an error rate close to 1.

fastest method in all cases, and MAFFT-linsi-add is the slowest when it can complete within the time limit. eMAFFTadd is usually slower than UPP-add but much faster than MAFFT-linsi-add (when it can run).

5.4 Experiment 4: Addressing Problem 2

Problem 2 is concerned with adding sequences into a growing alignment, an application of interest to evolutionary biologists. In this case, the added sequences may be full length, or they may be a mixture of lengths. For this experiment, we use the five CRW datasets. In each case, we select 100 sequences, either at random, the 100 shortest, or the 100 longest sequences. We then add these selected sequences into the reference alignment (i.e., as provided in [1], based on secondary structure) using each of the sequence-adding methods. Given the failures for MAFFT-linsi-add on large datasets from Experiment 3, we did not attempt to run MAFFT-linsi-add on the two largest datasets, 16S.3 and 16S.T, which have more than 6000 sequences.

Table 2 shows results without MAFFT-linsi-add across the five CRW datasets. Overall, errors are higher when adding the shortest or longest 100 sequences than random sequences for all methods. For all types of added queries, eMAFFTadd has the lowest alignment error and MAFFT-add the highest. eMAFFTadd also consistently has the lowest SPFN, and UPP-add has the lowest SPFP. In general, short sequences are the hardest to align for all of the methods, long sequences are also somewhat difficult, but the "random" sequences are relatively easy to

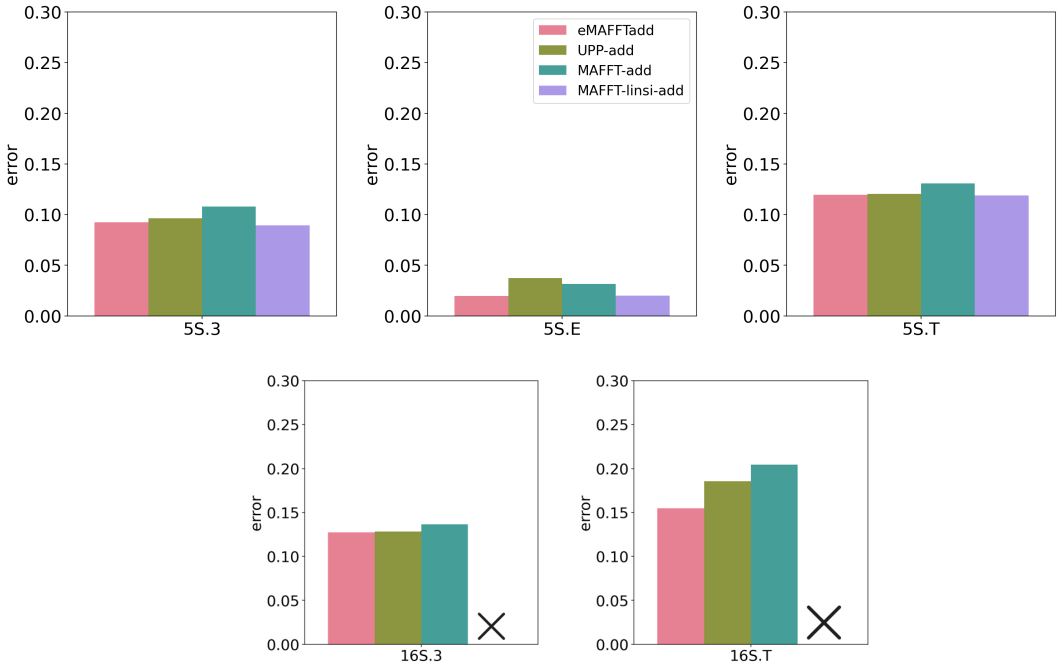

Figure 5: Experiment 3: Alignment error (average of SPFN and SPFP) of different alignment methods on query sequences of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add encountered out-of-memory issues on 16S.3 and 16S.T (marked as "X") and is only shown for 5S.3, 5S.E, and 5S.T.

align. Since the random sequences are a mixture of full-length, short, and long, this suggests that all the methods do well at adding full-length sequences. Another interesting observation is that eMAFFTadd has the lowest SPFN across all three categories, followed by UPP-add and then MAFFT-add. However, the gap between SPFN scores is large for eMAFFTadd over UPP-add and small between UPP-add and MAFFT-add.

Table 3 shows the same averages for those CRW datasets on which MAFFT-linsi-add finished (i.e., 5S.3, 5S.E, and 5S.T). This table thus allows us to see how MAFFT-linsi-add compares to other methods on those datasets on which it can run. One consistent trend in these data is that across all three types of query sequences (random, short, or long) and all criteria (SPFN, SPFP, or their average), MAFFT-add is the least accurate. Differences between the other methods are discernible when examining either short or long sequences, but there are only small differences when examining the random sequences. For both long and short query sequences, and considering SPFN or the average of SPFN and SPFP, eMAFFTadd is the most accurate, and the improvement on other methods is quite noticeable. Results are slightly different for SPFP, with UPP-add doing better than all other methods but with eMAFFTadd close behind. UPP-add substantially improves on MAFFT-linsi-add on the short sequences for both SPFN and SPFP. On long sequences, UPP-add has better average error than MAFFT-linsi-add, but this is achieved by a better SPFP and slightly

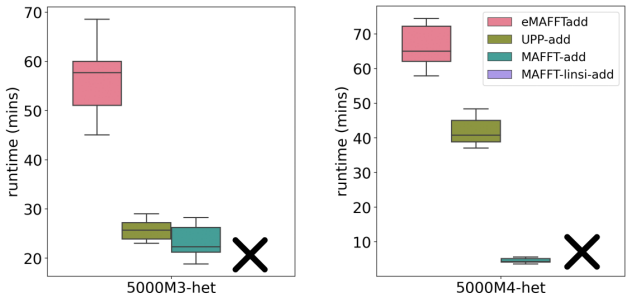

Figure 6: Experiment 3: Running time in minutes of different alignment methods on adding 4000 query sequences to 1000-taxon backbone alignments of 5000M3-het (left) and 5000M4-het (right), each with 10 replicates. Results for MAFFT-linsi-add are not shown because it failed to complete within 12 hours or encountered out-of-memory issues (marked as "X").

worse SPFN. Thus, the four methods show different responses to short and long sequences, but overall, eMAFFTadd provides the best accuracy. Finally, we note that, for all methods, the error rates are highest on the short sequences than on the long sequences. This indicates that short sequences present challenges for both types of sequence-addition methods (i.e., those using MAFFT techniques and those using HMMs to add sequences).

dataset	query type	method	SPFN	SPFP	average
CRW (5)	random 100	eMAFFTadd	**0.013**	0.013	**0.013**
		UPP-add	0.016	**0.011**	0.014
		MAFFT-add	0.017	0.014	0.016
	shortest 100	eMAFFTadd	**0.045**	0.039	**0.042**
		UPP-add	0.095	**0.033**	0.064
		MAFFT-add	0.109	0.089	0.099
	longest 100	eMAFFTadd	**0.034**	0.034	**0.034**
		UPP-add	0.077	**0.027**	0.052
		MAFFT-add	0.085	0.065	0.075

Table 2: Experiment 4: Average SPFN, SPFP, and their average (alignment error) over the five CRW datasets. We show results when adding 1) 100 randomly selected sequences, 2) 100 shortest sequences, or 3) 100 longest sequences in each dataset as queries. MAFFT-linsi-add is not shown because it is not run on 16S.3 and 16S.T. The lowest SPFN, SPFP, and their average are boldfaced.

6 Discussion

In this study, we presented eMAFFTadd, a new sequence-adding method that scales MAFFT-linsi --add to large datasets. Given an existing alignment (backbone alignment) and unaligned query sequences, it adds queries to the backbone alignment using MAFFT-linsi-add with a divide-and-conquer approach inspired by UPP [14, 18]. We showed that eMAFFTadd enables MAFFT-linsi-add on

dataset	query type	method	SPFN	SPFP	average
CRW (3)	random 100	eMAFFTadd	0.011	0.011	0.011
		UPP-add	0.012	**0.005**	**0.008**
		MAFFT-add	0.013	0.011	0.012
		MAFFT-linsi-add	**0.010**	0.010	0.010
	shortest 100	eMAFFTadd	**0.067**	0.055	**0.061**
		UPP-add	0.144	**0.042**	0.093
		MAFFT-add	0.172	0.138	0.155
		MAFFT-linsi-add	0.158	0.130	0.144
	longest 100	eMAFFTadd	**0.046**	0.045	**0.045**
		UPP-add	0.114	**0.031**	0.073
		MAFFT-add	0.128	0.096	0.112
		MAFFT-linsi-add	0.098	0.088	0.093

Table 3: Experiment 4: Average SPFN, SPFP, and their average (alignment error) over the three CRW datasets on which MAFFT-linsi-add completed (5S.3, 5S.E, and 5S.T). We show results when adding 1) 100 randomly selected sequences, 2) 100 shortest sequences, or 3) 100 longest sequences in each dataset as queries. The lowest SPFN, SPFP, and their average are boldfaced.

large datasets with at least 5000 sequences. It is also the most accurate among sequence-adding methods we tested under different sequence-adding scenarios.

It is easy to see why eMAFFTadd can scale to large datasets, as its design ensures that no problem is ever too big for MAFFT-linsi-add to complete. An explanation as to why eMAFFTadd is more accurate than MAFFT-add is straightforward: MAFFT-linsi-add is more accurate when it can run, and eMAFFTadd allows MAFFT-linsi-add to be used on these large datasets. Understanding why eMAFFTadd is more accurate than UPP-add is more interesting, but we offer a hypothesis. UPP-add operates by adding each query sequence independently, whereas MAFFT-linsi-add considers relationships it detects between query sequences and does not treat these additions independently; this may be key to why eMAFFTadd is more accurate than UPP-add. Finally, we note that there were cases where MAFFT-linsi-add and eMAFFTadd both ran, and we observed eMAFFTadd to be more accurate than MAFFT-linsi-add. The explanation for this is again somewhat subtle. While MAFFT-linsi is an exceptional alignment method, other studies (beginning with [9]) observed that MAFFT-linsi accuracy degraded on large datasets with high evolutionary diameters, and this observation led to the development of divide-and-conquer methods like SATé that improved accuracy by applying MAFFT-linsi to small diameter subsets of the sequence dataset. Thus, the improvement in accuracy for eMAFFTadd over MAFFT-linsi-add may result from a similar trend: restricting MAFFT-linsi-add to small diameter subsets may improve its accuracy.

It is also worth noting that where eMAFFTadd really excels is in its high recall (i.e., its ability to detect homologies). On all datasets in this study, eMAFFTadd has the lowest SPFN, and most of the time, the improvement on other methods is quite noticeable. We also note that eMAFFTadd has generally low SPFP scores, although UPP-add sometimes is better.

7 Conclusions

This study suggests many directions for future work. In particular, we only explored MAFFT-linsi-add rather than MAFFT-linsi-addfragments, which might provide better accuracy for adding short sequences. Improving the runtime for eMAFFTadd is also important, and variants that use faster sequence addition methods for the easy query sequences (i.e., full-length sequences) might improve speed without a loss of accuracy. Changes to the decomposition strategy might also lead to further improvements. More ambitiously, improvements might also be obtained by combining information from different sequence addition methods. Finally, not discussed here but of definite interest is whether these sequence addition methods are useful for distinguishing between remote homologs and non-homologs. For this to be achieved, it will be necessary to provide statistical support for the predicted alignment, which might be obtained by combining information from UPP-add and eMAFFTadd.

References

[1] Jamie J. Cannone, Sankar Subramanian, Murray N. Schnare, James R. Collett, Lisa M. D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V. Madabusi, Kirsten M. Müller, Nupur Pande, Zhidi Shang, Nan Yu, and Robin R. Gutell. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3(1):2, January 2002. doi:10.1186/1471-2105-3-2.

[2] Kodi Collins and Tandy Warnow. PASTA for proteins. *Bioinformatics*, 34(22):3939–3941, November 2018. doi:10.1093/bioinformatics/bty495.

[3] Robert D. Finn, Jody Clements, and Sean R. Eddy. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research*, 39(Web Server issue):W29–W37, July 2011. doi:10.1093/nar/gkr367.

[4] William Fletcher and Ziheng Yang. INDELible: A Flexible Simulator of Biological Sequence Evolution. *Molecular Biology and Evolution*, 26(8):1879–1888, August 2009. doi:10.1093/molbev/msp098.

[5] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Å½Ãdek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, August 2021. Number: 7873 Publisher: Nature Publishing Group. doi:10.1038/s41586-021-03819-2.

[6] Kazutaka Katoh and Martin C. Frith. MAFFT – a multiple alignment program for amino acid or nucleotide sequences. https://mafft.cbrc.jp/alignment/software/addsequences.html. Accessed: 2022-05-20.

[7] Kazutaka Katoh and Martin C Frith. Adding unaligned sequences into an existing alignment using MAFFT and LAST. *Bioinformatics*, 28(23):3144–3146, 2012. Publisher: Oxford University Press.

[8] Kazutaka Katoh and Daron M. Standley. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. *Molecular Biology and Evolution*, 30(4):772–780, April 2013. doi:10.1093/molbev/mst010.

[9] Kevin Liu, Sindhu Raghavan, Serita Nelesen, C. Randal Linder, and Tandy Warnow. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. *Science*, 324(5934):1561–1564, June 2009. URL: https://science.sciencemag.org/content/324/5934/1561, doi:10.1126/science.1171243.

[10] S. Mirarab, N. Nguyen, and T. Warnow. SEPP: SATé-Enabled Phylogenetic Placement. In *Biocomputing 2012*, pages 247–258. WORLD SCIENTIFIC, November 2011. URL: https://www.worldscientific.com/doi/abs/10.1142/9789814366496_0024, doi:10.1142/9789814366496_0024.

[11] Siavash Mirarab, Nam Nguyen, Sheng Guo, Li-San Wang, Junhyong Kim, and Tandy Warnow. PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences. *J Comput Biol*, 22(5):377–386, May 2015. doi:10.1089/cmb.2014.0156.

[12] Siavash Mirarab and Tandy Warnow. FASTSP: linear time calculation of alignment accuracy. *Bioinformatics*, 27(23):3250–3258, December 2011. doi:10.1093/bioinformatics/btr553.

[13] David A Morrison. Multiple sequence alignment for phylogenetic purposes. *Australian Systematic Botany*, 19(6):479–539, 2006.

[14] Nam-phuong D. Nguyen, Siavash Mirarab, Keerthana Kumar, and Tandy Warnow. Ultra-large alignments using phylogeny-aware profiles. *Genome Biology*, 16(1):124, June 2015. doi:10.1186/s13059-015-0688-z.

[15] Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin. FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments. *PLOS ONE*, 5(3):e9490, March 2010. doi:10.1371/journal.pone.0009490.

[16] Bruce A Shapiro, Yaroslava G Yingling, Wojciech Kasprzak, and Eckart Bindewald. Bridging the gap in RNA structure prediction. *Current Opinion in Structural Biology*, 17(2):157–165, 2007.

[17] Chengze Shen, Minhyuk Park, and Tandy Warnow. WITCH: Improved Multiple Sequence Alignment Through Weighted Consensus Hidden Markov Model Alignment. *Journal of Computational Biology*, May 2022. Publisher: Mary Ann Liebert, Inc., publishers. doi:10.1089/cmb.2021.0585.

[18] Chengze Shen, Paul Zaharias, and Tandy Warnow. MAGUS+eHMMs: improved multiple sequence alignment accuracy for fragmentary sequences. *Bioinformatics*, 38(4):918–924, February 2022. doi:10.1093/bioinformatics/btab788.

[19] Vladimir Smirnov and Tandy Warnow. MAGUS: Multiple sequence Alignment using Graph clUStering. *Bioinformatics*, 37(12):1666–1672, June 2021. doi:10.1093/bioinformatics/btaa992.

[20] Vladimir Smirnov and Tandy Warnow. Phylogeny Estimation Given Sequence Length Heterogeneity. *Systematic Biology*, 70(2):268–282, March 2021. doi:10.1093/sysbio/syaa058.

[21] Jeet Sukumaran and Mark T. Holder. DendroPy: a Python library for phylogenetic computing. *Bioinformatics*, 26(12):1569–1571, 04 2010. doi:10.1093/bioinformatics/btq228.

A Commands for software

1. Generating MAGUS backbone (we used MAGUS GitHub version committed on April 5th 2021):

```
$ python3 magus.py --recurse false -np 16  \
    -i [unaligned backbone sequences] -d [outdir] \
    -o [output alignment]
```

2. Generating FastTree2 backbone tree (on nucleotide, v2.1 multi-threaded version):

```
$ FastTreeMP -nt -gtr [backbone alignment] > [backbone tree]
```

3. Running UPP-add (UPP version v4.5.0):

```
$ python3 run_upp.py -x 16 -s [query sequences path] \
    -a [backbone alignment] -t [backbone tree] -p [tempdir] \
    -d [outdir] -o [job name]
```

4. Running MAFFT/MAFFT-linsi --add:

```
$ [mafft/mafft-linsi] --quiet --thread 16 --add [query sequences] \
    [backbone alignment] > [output alignment]
```

B Dataset Generation

Condition	l **(tree scale)**	r **(indel rate)**
5000M2	30	4.4×10^{-4}
5000M3	20	3.3×10^{-4}
5000M4	5	1.32×10^{-3}

Table 4: Parameters for generating the 5000M-het series data, where l is the tree scale parameter, defining the maximimum path-length in the non-ultrametric model tree, and r defines the indel rate.

We present details for the generation of our new simulated conditions below. Our simulation parameters are based on those of the 1000M series of the ROSE simulated dataset [9]. We uploaded all of our INDELible control files generating this data to `https://github.com/ThisBioLife/5000M-234-het` to allow easy reproduction.

B.1 Model trees

Our model trees were generated by a two-step process. We first generated random 5000-species birth-death trees (one tree per replicate) using the DendroPy [21] `treesim.birth_death_tree` function, setting the birth rate initially at 1 with the standard deviation of the change to the birth rate set to 0.2 (see the documentation on this Dendropy function on the birth-death process and the

implication of this parameter), with a zero death rate. Then, taking only the topology of the tree generated in the prior step, we instructed INDELible to assign branch lengths to the tree s.t. the resulting tree is both non-ultrametric and has a maximum path-length l. We vary l to control the scale of the tree and hence the rate of evolution of the condition. The choice of l across conditions can be seen in Table 4.

B.2 Indel length distribution & indel rates

Our model for sequence length heterogeneity assumes a base ("short indel") distribution of indel lengths. In our case we simply took the "medium" length distribution from the "M"-series ROSE simulated datasets as the short indel distribution. Given an indel event, with probability $p = 0.85$, it draws its length from the short indel distribution. Otherwise, it draws its length from a long indel distribution, in our case set to $\mathrm{NB}(130, 0.5)$. The indel length distribution is thus equivalently a mixture distribution of the short indel distribution (prior probability 0.85) and the long indel distribution (prior probability 0.15). We directly computed the probability mass function (PMF) of this mixture distribution, truncated the PMF, and fed the truncated PMF to INDELible as part of the input. The truncated PMF can be found alongside the uploaded control files in the Github repository linked.

Similar to the original ROSE dataset, we also varied the indel rates across conditions, with the choices for the indel rate r shown in Table 4. The indel rates were chosen analogous to the original indel rates of the ROSE dataset.

B.3 Other parameters

The rest of the parameters (GTR+Γ parameters and the initial sequence length) were chosen to be the same as the original ROSE dataset and can be found in the uploaded control files.

C MAFFT-linsi-add scalability issue on 5000-taxon datasets

Our runtime environment is 16 cores, 64 GB memory, and 12-hour runtime limit.

We tried running MAFFT-linsi --add (MAFFT-linsi-add) on our training 5000M2 datasets, where for each replicate, we added 4000 query sequences to a 1000-taxon full-length backbone alignment. We encountered either out-of-memory issues (64 GB memory limit) or crashes. We also conducted an experiment on exploring the scalability of MAFFT-linsi-add by altering the number of queries (Experiment 1).

The out-of-memory error message looks like the following:

```
slurmstepd: error: Detected 1 oom-kill event(s) in
StepId=5376434.batch cgroup. Some of your processes may have been
killed by the cgroup out-of-memory handler.
```

The crash error message looks like the following:

```
Command exited with non-zero status 1
```

D Additional Figures

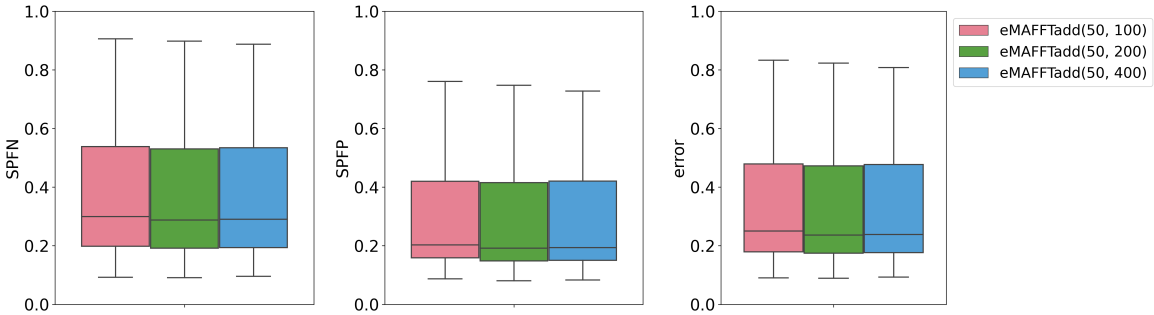

Figure 7: Experiment 2: SPFN (left), SPFP (middle), and alignment error (right, the average of SPFN and SPFP) of eMAFFTadd with various settings for adding all query sequences to a 1000-taxon alignment of 5000M2-het (10 replicates). eMAFFTadd variants are marked as eMAFFTadd(50, u), for which $u = \{100, 200, 400\}$ controls the maximum size of sub-alignments to use for MAFFT-linsi-add. All metrics are calculated on query sequences only.

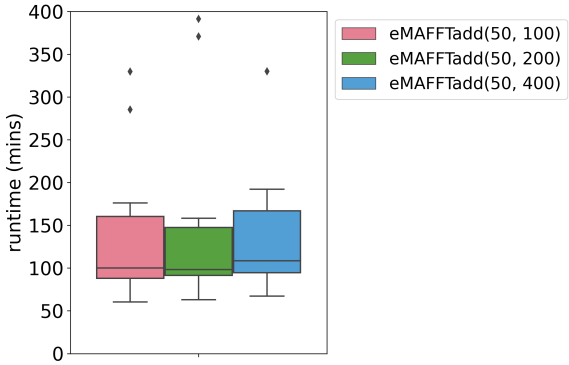

Figure 8: Experiment 2: Runtime in minutes for eMAFFTadd with varying u, $u = [100, 200, 400]$, when adding 4000 query sequences to 1000-taxon backbone alignments of 5000M2-het (10 replicates).

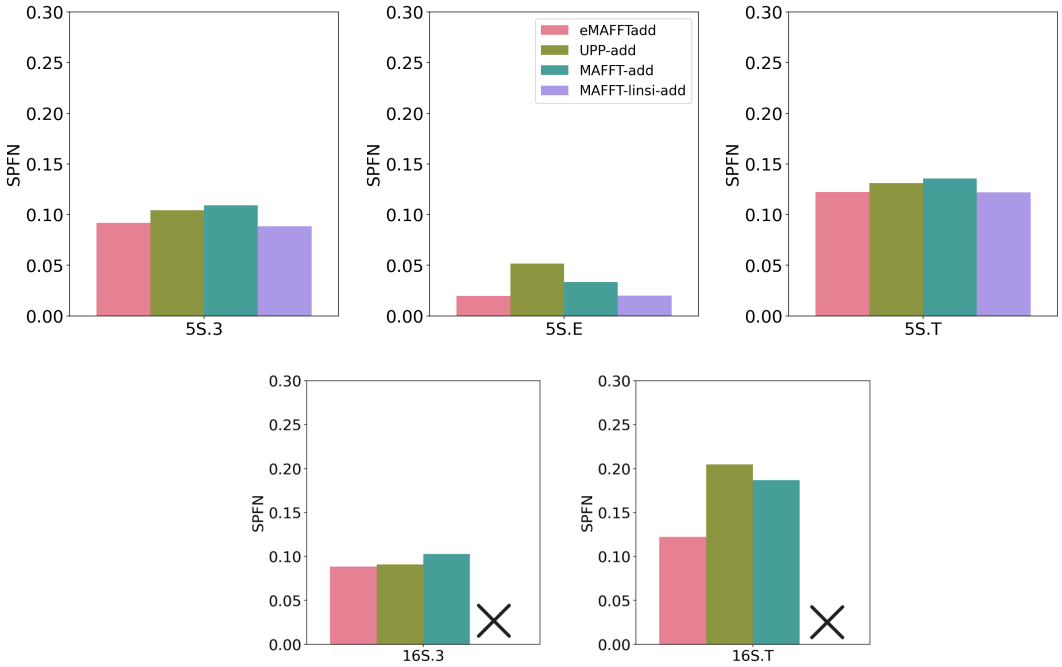

Figure 9: Experiment 3: SPFN of different alignment methods on query sequences of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add encountered out-of-memory issues on 16S.3 and 16S.T (marked as "X") and is only shown for 5S.3, 5S.E, and 5S.T.

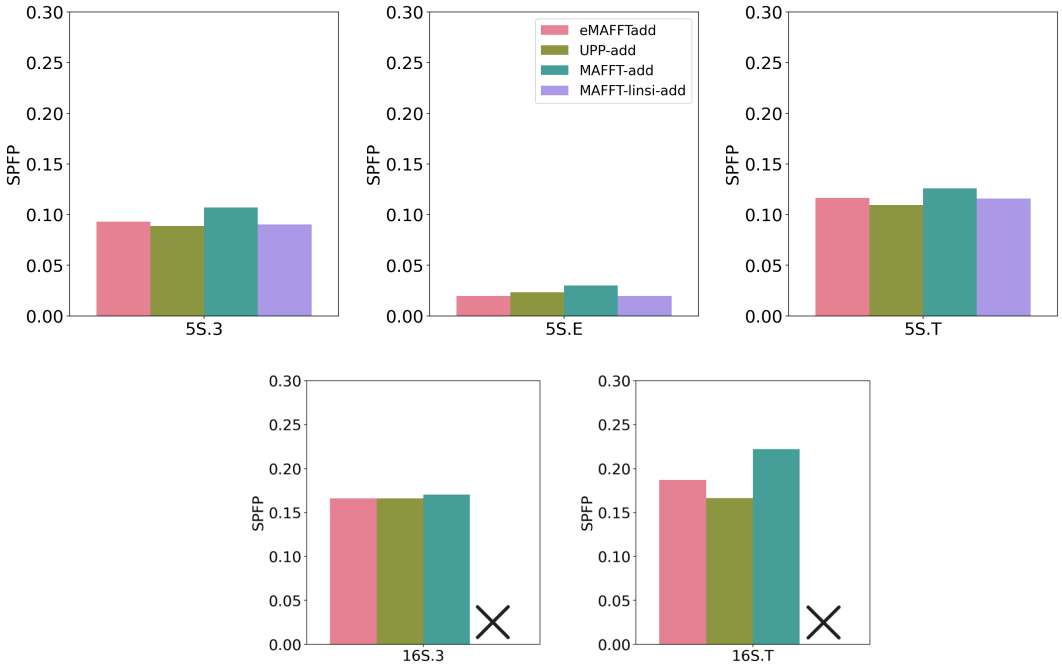

Figure 10: Experiment 3: SPFP of different alignment methods on query sequences of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add encountered out-of-memory issues on 16S.3 and 16S.T (marked as "X") and is only shown for 5S.3, 5S.E, and 5S.T.

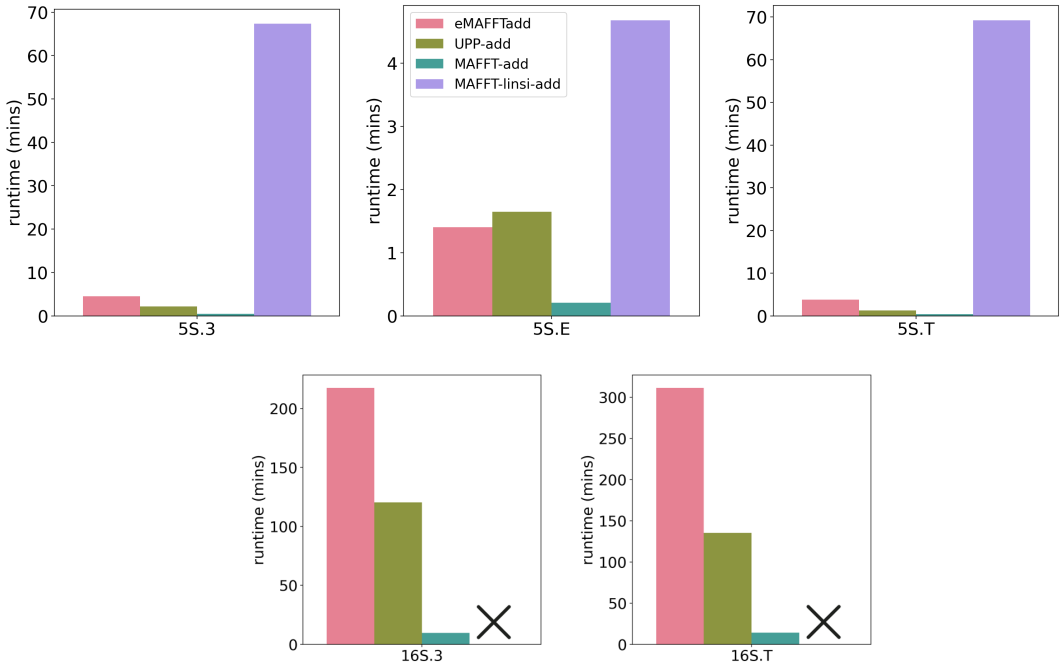

Figure 11: Experiment 3: Running time in minutes of different alignment methods on adding 4000 query sequences to 1000-taxon backbone alignments of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add encountered out-of-memory issues on 16S.3 and 16S.T, and is not shown (marked as "X").

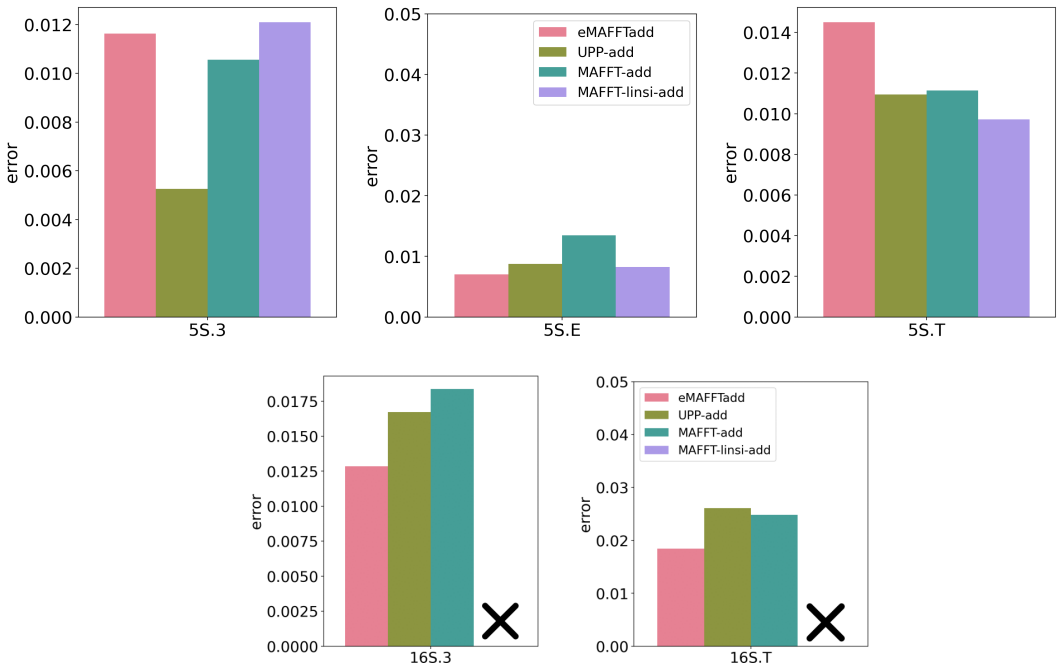

Figure 12: Experiment 4: Alignment error (average of SPFN and SPFP) of different alignment methods when adding **100 random sequences** to the reference backbone alignment of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add is not run on both 16S datasets since it does not scale in Experiment 3 (marked as "X").

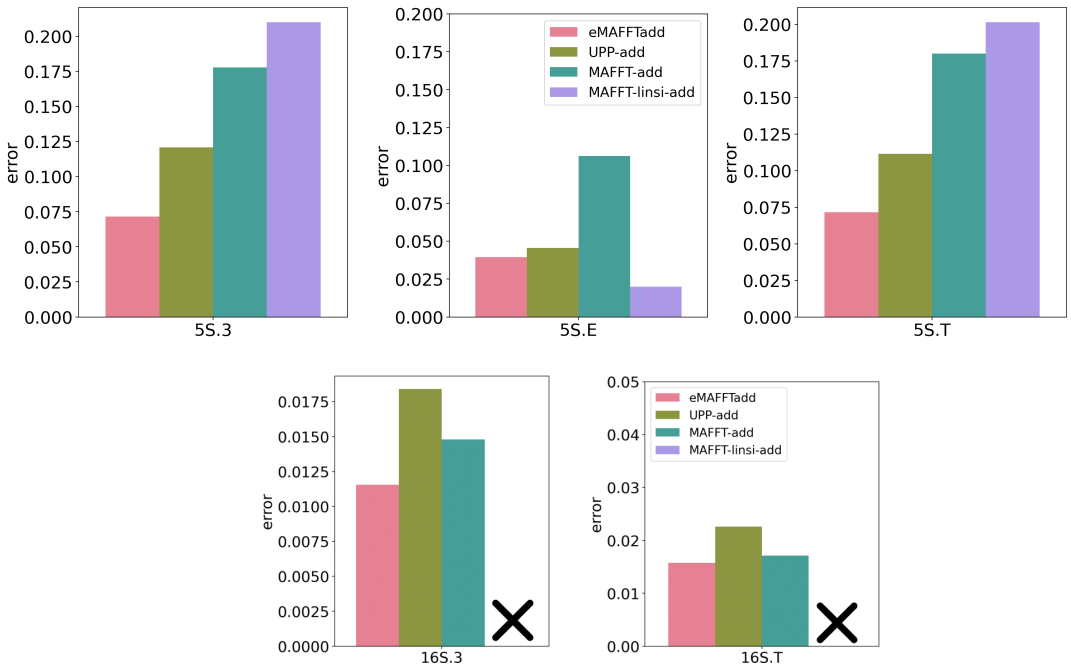

Figure 13: Experiment 4: Alignment error (average of SPFN and SPFP) of different alignment methods when adding **the shortest 100 sequences** to the reference backbone alignment of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. MAFFT-linsi-add is not run on both 16S datasets since it does not scale in Experiment 3 (marked as "X").

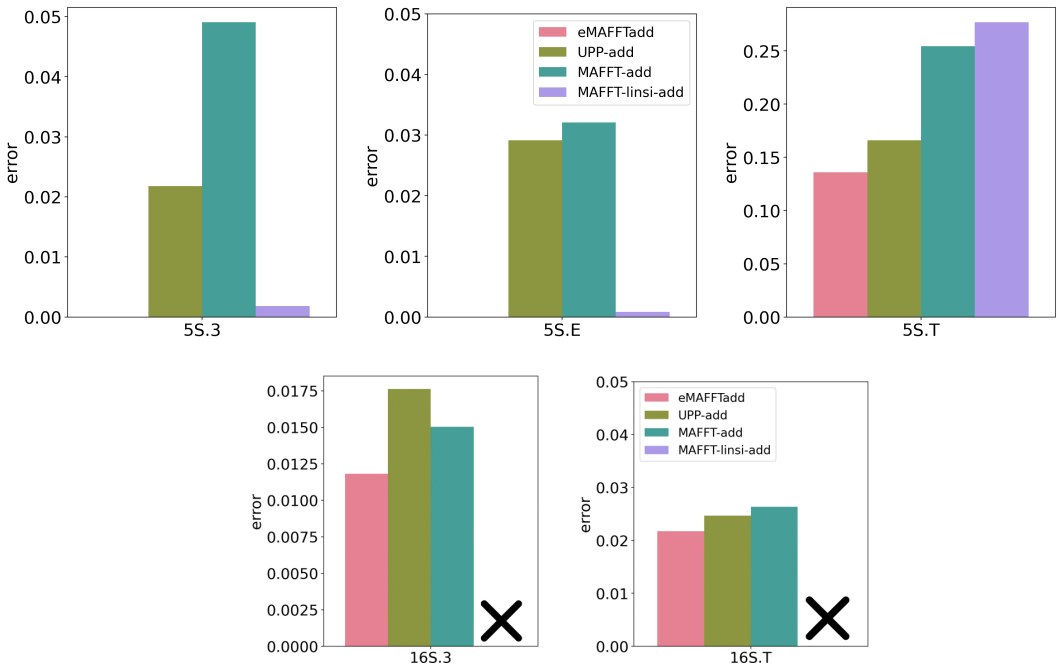

Figure 14: Experiment 4: Alignment error (average of SPFN and SPFP) of different alignment methods when adding **the longest 100 sequences** to the reference backbone alignment of CRW 5S.3, 5S.E, 5S.T, 16S.3, and 16S.T datasets. eMAFFTadd has zero error on 5S.3 and 5S.T. MAFFT-linsi-add is not run on both 16S datasets since it does not scale in Experiment 3 (marked as "X").