UPP2: Fast and Accurate Alignment Estimation of Datasets with Fragmentary Sequences

Minhyuk Park[1] Stefan Ivanovic[1] Gillian Chu[1]
Chengze Shen[1] Tandy Warnow[1*]

[1]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820
[*]To whom correspondence should be addressed.

Abstract

Motivation: Multiple sequence alignment (MSA) is a basic step in many bioinformatics pipelines. However, achieving highly accurate alignments on large datasets, especially those with sequence length heterogeneity, is a challenging task. UPP (Ultra-large multiple sequence alignment using Phylogeny-aware Profiles) is a method for MSA estimation that builds an ensemble of Hidden Markov Models (eHMM) to represent an estimated alignment on the full length sequences in the input, and then adds the remaining sequences into the alignment using selected HMMs in the ensemble. Although UPP provides good accuracy, it is computationally intensive on large datasets.

Results: We present UPP2, a direct improvement on UPP. The main advance is a fast technique for selecting HMMs in the ensemble that allows us to achieve the same accuracy as UPP but with greatly reduced runtime. We show UPP2 produces more accurate alignments compared to leading MSA methods on datasets exhibiting substantial sequence length heterogeneity, and is among the most accurate otherwise.

Availability: https://github.com/gillichu/sepp

Contact: warnow@illinois.edu

1 Introduction

Multiple sequence alignment is a fundamental bioinformatics task, and producing accurate alignments can have profound impact in many downstream analyses such as phylogeny inference and protein structure inference (Blackburne and Whelan, 2013; Söding, 2004; Rao *et al.*, 2021). Because of the significant interest in alignment estimation, many alignment methods have been developed (e.g., MUSCLE (Edgar, 2004), PRANK (Löytynoja and Goldman, 2005), BAli-Phy (Suchard and Redelings, 2006), Clustal Omega (Sievers *et al.*, 2011), MAFFT (Katoh and Standley, 2013), PASTA (Mirarab *et al.*, 2015), MAGUS (Smirnov

and Warnow, 2021b), and regressive T-COFFEE (Garriga *et al.*, 2019)). However, accurate alignment is still challenging under some conditions. For example, large datasets (with many thousands of sequences) can be difficult to align with high accuracy and also present substantial computational challenges (Nguyen *et al.*, 2015; Mirarab *et al.*, 2015; Smirnov, 2021). The difficulty in aligning datasets that are highly heterogeneous due to high rates of evolution has also been documented (Liu *et al.*, 2009), but several methods (largely employing divide-and-conquer) have been able to achieve good accuracy in such conditions (e.g., PASTA (Mirarab *et al.*, 2015) and MAGUS (Smirnov and Warnow, 2021b)). Sequence length heterogeneity introduces another challenge for alignment estimation, and one that is relatively less studied (Nguyen *et al.*, 2015; Shen *et al.*, 2021).

UPP (Ultra-large alignments using Phylogeny-aware Profiles) (Nguyen *et al.*, 2015)) is a multiple sequence alignment method that was specifically designed to provide good accuracy on datasets with substantial sequence length heterogeneity, while maintaining scalability on large datasets. UPP operates in three basic stages: first, it extracts and aligns a subset of the sequences it deems to be full-length; second, it builds an ensemble of Hidden Markov Models (HMMs) (Durbin *et al.*, 1998) on the alignment of the full-length sequences; and third, it uses the ensemble to align the remaining sequences. This third step is often the bottleneck in terms of runtime. Specifically, for each additional sequence that needs to be aligned, the HMM with the highest bit-score is selected from the ensemble and is used to add the sequence into the alignment. By design, the first two steps are reasonably fast, but the third step requires an all-against-all comparison of the remaining sequences against the HMMs in the ensemble. Thus, the runtime of UPP can be prohibitively high when there are many sequences that are not full-length and when the ensemble contains many HMMs.

In the last year, modifications to UPP to improve its accuracy and theoretical foundation have been explored. The default for UPP (Mirarab, 2022) uses PASTA to align the backbone sequences. However, Shen *et al.* (2021) showed that alignment accuracy was improved by using MAGUS instead of PASTA to compute the backbone alignment. Another potential weakness in the original UPP approach is the use of the bit score to select the single HMM to align the query sequence. A modification to this technique was presented in Shen *et al.* (2022), which replaces the use of the bit score by another calculation that more accurately reflects the probability that the given HMM generated the query sequence. Thus, the current recommended setting for UPP uses MAGUS for the backbone alignment and selects the "best" HMM from the ensemble based on the statistical calculation presented in Shen *et al.* (2022). However, the version in the github site (Mirarab, 2022) still uses PASTA for the backbone and selects the best HMM based on raw bit-scores.

These modifications aimed to improve accuracy rather than runtime, and UPP has remained computationally intensive as a result of its all-against-all algorithmic design. Here we present UPP2 (available in open source form at https://github.com/gillichu/sepp), a modification to UPP that is designed to reduce its runtime and improve its scalability to large sequence datasets. The

main modification we use is a replacement of the all-against-all comparison of query sequences and HMMs by a much smaller number of comparisons, so that each query sequence is scored against a logarithmic number of HMMs instead of against all the HMMs. This change dramatically reduces the runtime without hurting accuracy, as our study on simulated and biological datasets establishes.

2 UPP2

2.1 The UPP algorithm

Recall that UPP operates in several stages. In the first stage, it computes a backbone alignment and backbone tree on a subset of the input sequences, in the second stage it builds an ensemble of profile HMMs on the backbone alignment, and in the third stage it uses the ensemble to add all the remaining sequences into the backbone alignment using commands from HMMER (Eddy, 2011) (hmmbuild, hmmsearch, and hmmalign). Here we provide some additional details.

For Stage 1, by default, UPP will select up to 1000 sequences to include in its backbone, and these sequences are selected at random from the set of sequences within 25% in length of the median length sequence. The alignment is built using a selected "base method", with PASTA the original technique and now MAGUS the recommended technique.

For Stage 2, UPP computes a set of subset alignments by hierarchically decomposing the backbone tree at a centroid edge (i.e., an edge that splits the leaf set into two roughly equal sizes) until all the subtrees are at most size z, where z is an input to UPP. UPP builds an HMM on each set created during this decomposition, including the full set, thus producing a collection of HMMs that we refer to as the "ensemble of HMMs" (eHMM) for the backbone alignment. In the initial version of UPP, z was set to 10. Some studies (Mirarab *et al.*, 2012) that developed eHMMs for other purposes have suggested that smaller values (e.g., $z = 2$) might improve accuracy, but a more recent study exploring this question for alignment estimation Shen *et al.* (2021) has found otherwise.

For Stage 3, UPP adds every additional sequence (i.e., ones that are not in the backbone) into the backbone alignment. These additional sequences are referred to as "query sequences" and are added as follows. For each query sequence, *hmmsearch* is used to find the HMM that returns the highest bit-score. Then, each query sequence is added into the subset alignment used to construct the selected HMM using *hmmalign*. Since the subset alignments are induced by the backbone alignment, this also means the query sequence can be added into the backbone alignment as well. The addition of the query sequence into the backbone alignment defines an "extended alignment". Finally, the extended alignments from the different query sequences are merged together using transitivity, thus producing a final alignment containing all the sequences.

2.2 Adjusted bit-scores

A bit-score represents the log likelihood ratio of a query sequence being emitted by an HMM to the likelihood of a query sequence being emitted by a null HMM. However, the raw bit-score does not correspond to the probability that the query sequence is generated by the selected HMM, as this specific question depends also on the number of sequences used to build the HMM.

To address this, a modification to the use of bit-scores, called "adjusted bit-scores", is presented in Shen *et al.* (2022). We provide the adjusted bit-score formula here; the derivation of the formula is provided in the Supplementary Materials and also in Shen *et al.* (2022).

Suppose $BS(H_i, q)$ is the bit-score of query sequence q for the HMM H_i, s_i is the number of sequences that were used to build H_i, s_j is the number of sequences that were used to build H_j, and h is the number of HMMs in the ensemble. The adjusted bit-score, denoted by BS_{adj}, is then given by

$$BS_{adj}(H_i, q) = \frac{1}{\sum_{j=1}^{h} 2^{BS(H_j,q) - BS(H_i,q) + \log_2 \frac{s_j}{s_i}}} \tag{1}$$

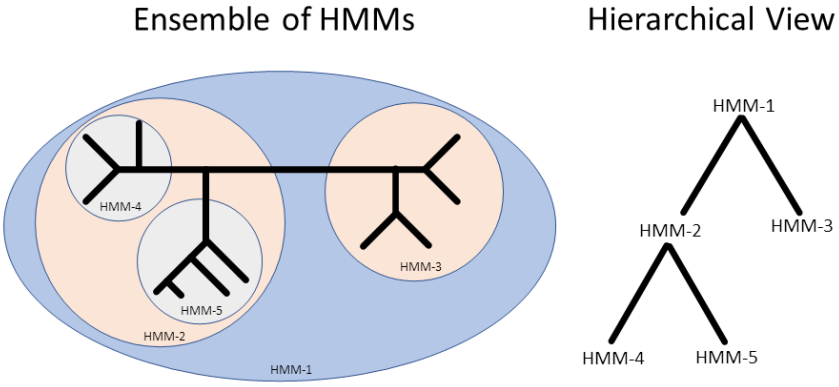
As shown in Shen *et al.* (2022), adjusted bit-scores are always between 0 and 1, and can be interpreted as the probability that the given HMM generates the given query sequence. In particular, the sum, across all the HMMs in the ensemble, is always 1.

2.3 Improving speed

Due to its hierarchical decomposition strategy, UPP produces many HMMs, all of which have to be compared against every single query sequence. This quickly presents scalability issues in several cases: as the size of backbone increases, as z (which defines the decomposition stopping rule) decreases, or as the number of query sequences increases. However, Stage 2 defines a hierarchy of HMMs based on their sequence sets, so that the set of HMMs forms a rooted tree. We propose two strategies ("Hierarchical" and "EarlyStop") based on this hierarchy to speed up the search: Hierarchical and EarlyStop (Figure 1).

Hierarchical To select an HMM for a given query sequence q, we start at the root HMM and we compute its adjusted bit-score given q. We then evaluate the children HMMs and descend down into the subtree that has the larger adjusted bit-score (randomly selecting one in the case of a tie). The process continues until a leaf HMM is reached. The HMM with the largest adjusted bit-score (i.e., the HMM deemed the most likely to have emitted the query sequence) encountered during the traversal then becomes selected HMM for the query sequence. Note that this strategy evaluates at most two HMMs per level in the tree. In the case of a tie, the HMM that comes first in a pre-order traversal is chosen.

EarlyStop We follow the same basic strategy as Hierarchical. However, the process stops descending down the subtree in the hierarchical search process if



HMM	Bit-score	Adjusted Bit-score
HMM-1	6	0.518
HMM-2	4	0.082
HMM-3	7	0.376
HMM-4	3	0.018
HMM-5	1	0.006

Method	HMM(s) Queried	HMM Chosen
UPP	1,2,3,4,5	3
UPP+adj	1,2,3,4,5	1
UPP+adj+HS	1,2,3	1
UPP+adj+EarlyStop	1,2,3	1

Figure 1: **UPP and UPP2 Search Strategies (Toy Example)** Here we show an ensemble of HMMs with HMMs queried and chosen for different methods. UPP and UPP2 by default search through every HMM but use different criteria (raw bit-scores or adjusted bit-scores, respectively). UPP will choose HMM-1 since HMM-1 has the highest bit-score while UPP2 will choose HMM-2 since HMM-2 has the highest adjusted bit-score. UPP2-Hierarchical will start at HMM-1 and descend down the subtree with the highest adjusted bit-score. UPP2-EarlyStop will descend down the subtree with the highest adjusted bit-score and stop once all immediate children HMMs have worse adjusted bit-scores than the current best HMM. In all cases, the HMM with the highest adjusted bit-score encountered during the traversal is chosen.

both of the two children HMMs have lower adjusted bit-scores, and are therefore considered less likely to have emitted the query sequence, than the current best HMM (hence the name "EarlyStop").

3 Experimental Study

Overview. We performed two experiments, one for designing UPP2 (Experiment 1) and one for evaluating UPP2 in comparison to leading alignment methods (Experiment 2). Experiment 1 was performed on a small set of "training datasets" and Experiment 2 was performed on a larger set of "testing datasets". Accuracy in these alignments was evaluated with respect to SPFN (sum-of-pairs false negative) and SPFP (sum-of-pairs false positive) error rates. We also evaluated the total runtime for all methods as well as the runtime restricted to Stage 3 for UPP variants, which is the only stage in which UPP variants differ from each other.

Alignment methods. We evaluated possible variants of UPP2, which differ in terms of the backbone alignment method (PASTA or MAGUS), the use of raw or adjusted bit-scores, the stopping condition (i.e., how z is set), and whether the all-against-all comparisons are performed or one of the two faster search strategies is used. Recall that the original version of UPP uses PASTA backbones, raw bit-scores, sets $z = 10$, and performs all-against-all comparisons to find the best HMM for each query sequence.

We explore the following variants of UPP2 (indicating how they differ from the original version of UPP below). Each of these versions is described with the PASTA backbone; when used with the MAGUS backbone, the name of the method is modified by the inclusion of "(MAGUS)".

- *UPP+adj*: UPP+adj differs from the original UPP by using adjusted bit-scores.
- *UPP+adj-Hierarchical*: Identical to UPP+adj except that it uses the hierarchical search strategy.
- *UPP+adj-EarlyStop*: Identical to UPP+adj except that it uses the EarlyStop search strategy.

We also evaluated the following leading alignment methods:

- **MUSCLE** (3.8.31), limited to 2 iterations.
- **Clustal Omega** (1.2.4), used in its default mode.
- **T-COFFEE** (13.45.0.4846264), used in regressive mode.
- **MAGUS**, (Commit on 4/5/21, commit ID in supplement) used in its default mode, which is with recursion in the newest version of MAGUS (Smirnov and Warnow, 2021b; Smirnov, 2021).

- **PASTA** (v1.9.0), used in its default mode.
- **MAFFT** (7.487), with the *linsi* mode used for datasets of size at most 1000 and the *auto* mode used for larger datasets.
- **UPP** (4.5.1), where we set $z = 2$ but otherwise use it in default mode, as specified in its github page (Mirarab, 2022).

Datasets. We used both biological and simulated datasets for the experiments, separating them into the training datasets (used in Experiment 1) and the testing datasets (used in Experiment 2). We had fragmentary versions of the datasets; for these datasets, the suffix "HF" denotes high fragmentary datasets, which are constructed by taking the original dataset and making half of the sequences 25% of the original median sequence length. The fragmentation process is explained in full detail in Smirnov and Warnow (2021c). The empirical statistics (i.e., number of sequences, average sequence length, percent of the reference alignment occupied by gaps, and average and maximum p-distance) for these datasets are provided in the Supplementary Materials Table S1.

The ROSE simulated datasets, introduced in Liu *et al.* (2009), are 1000-sequence datasets with varying gap lengths, which are denoted by "S" for short gap lengths, "M" for medium gap lengths, and "L" for long gap lengths. We used 1000S1 through 1000S5, 1000M1 through 1000M5, and 1000L1 through 1000L5 as well as their high fragmentary counterparts 1000S1-HF through 1000S5-HF, 1000M1-HF through 1000M5-HF, and 1000L1-HF through 1000L5-HF. 1000M1 and 1000M1-HF were using for training while the other model conditions were reserved for testing. These datasets range in alignment difficulty as a result of the evolutionary rate, as reflected in their average p-distance (i.e., average normalized Hamming distance between any two sequences). The most difficult model conditions are 1000M1, 1000S1, and 1000L1, and the easiest ones are 1000S5, 1000M4, 1000M5, and 1000L5. Comparing the p-distances, we see that the difficult model conditions have average p-distances at least 69% and the easiest model conditions have average p-distances below 50%.

RNASim datasets are created by sampling from the RNASim million sequence dataset, originally created by Guo *et al.* (2009) and studied in Mirarab *et al.* (2015). We used the same 1000-sequence RNASim datasets as Smirnov and Warnow (2021c), which are published at Smirnov and Warnow (2021a); these are the RNASim1000 and RNASim1000-HF. RNASim1000 and RNASim1000-HF datasets. The RNASim datasets have average p-distance of 41%.

The 16S datasets are biological datasets based on the 16S gene. The 16S.BALL, 16S.3, and 16S.T datasets were used in the study by Liu *et al.* (2011) and are available at Mirarab (2017). These datasets have reference alignments based on secondary structure (Cannone *et al.*, 2002). 16S.3 has average p-distance of 31.5%, 16S.T has average p-distance of 34.5%, and 16S.B.ALL has average p-distance of 21.0%.

For the training datasets (Experiment 1), we used two model conditions, 1000M1 (one of the model conditions from the ROSE simulated datasets with

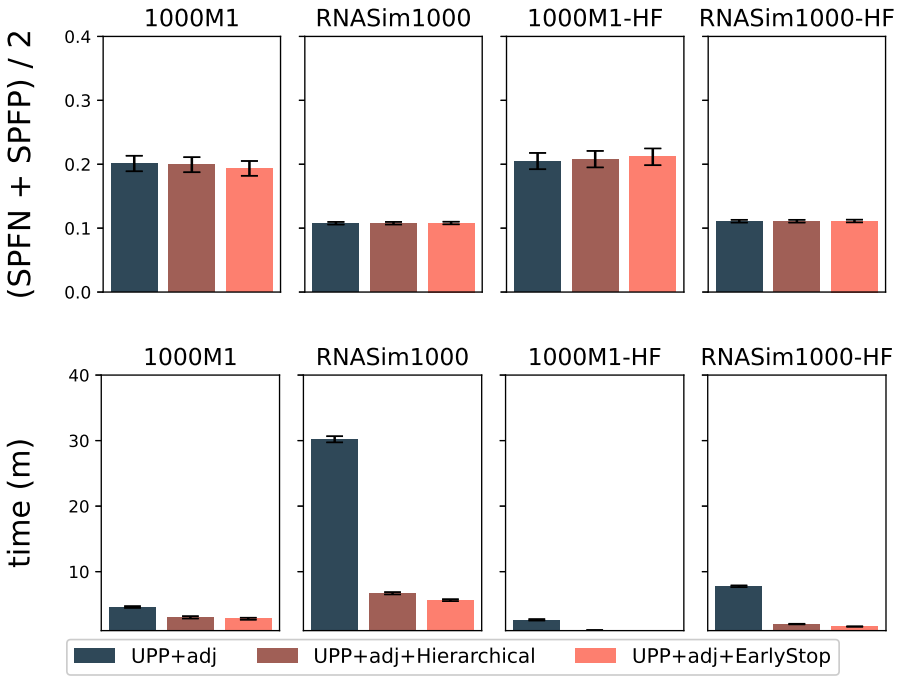


Figure 2: **Experiment 1:** Impact of Heuristic Search and Early Stop on UPP(PASTA)+adj alignment error and runtime. UPP+adj uses the PASTA backbone alignment on full-length sequences, sets $z = 2$, and uses adjusted bitscores. The runtime reported here does not include the time to compute the backbone alignment and tree. The means are shown with error bars indicating standard error for alignment error and standard deviation for running time.

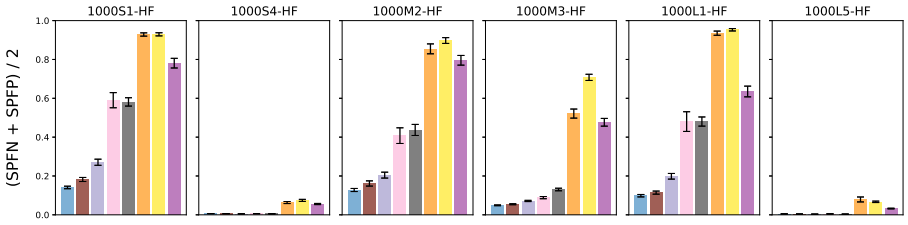
high rates of evolution) and RNASim1000. For each model condition, we explored full-length versions and HF versions. We used the remaining datasets as the testing datasets (Experiment 2).

Computational Resources. All experiments except MAFFT runs on the large biological datasets (16S.B.ALL, 16S.T, and 16S.3) and UPP runs on ROSE high fragmentary datasets were done on Blue Waters (Bode *et al.*, 2013) with every method receiving 16 threads on a dedicated node. MUSCLE does not have multi-threaded versions and was unable to take advantage of the core count. ROSE high fragmentary dataset UPP runs and 16S dataset MAFFT runs were done on the Campus Cluster with 16 threads. All methods were limited to a maximum of 7 days of wall-time and enough memory to complete their analyses.

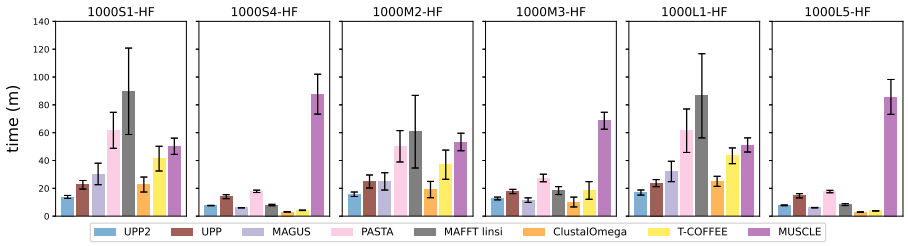
Alignment Error. We used FastSP (1.7.1) for calculating SPFN and SPFP rates of estimated alignments against the reference alignments (Mirarab and Warnow, 2011), defined as follows. SPFN refers to "sum-of-pairs false negatives", and is the number of the pairwise homologies found in the reference alignment but not in the estimated alignment, while SPFP refers to "sum-of-pairs false positives" and is the number of pairwise homologies found in the estimated alignment but not in the reference alignment. These are normalized by the number of homologies in the reference alignment or estimated alignment, respectively, to produce the SPFN and SPFP error rates.

Experiment 1 Overview. Experiment 1 explored the impact of decomposition stopping criterion (i.e., value for z), use of adjusted bit-scores, and new search strategies (Hierarchical and EarlyStop) for UPP2. We additionally evaluate the impact of using MAGUS backbone alignments instead of PASTA backbone alignments for use in UPP2. . We used 1000M1, RNASim1000, 1000M1-HF, and RNASim1000-HF. For 1000M1 and RNASim1000, 500 full length backbone sequences were selected by UPP. 1000M1-HF and RNASim1000-HF, the full-length sequences are easily identified, and that information is provided to UPP.

Experiment 2 Overview. We used Experiment 1 to specify how best to run the new algorithm, and refer to this variant as "UPP2". Experiment 2 compared UPP2, UPP, MAGUS, PASTA, MAFFT (using *linsi* for datasets with at most 1000 sequences and *auto* otherwise), Clustal-Omega, regressive T-COFFEE, and MUSCLE. Experiment 2a examined results on simulated datasets with fragmentary sequences, and Experiment 2b examined results on biological datasets. Finally, Experiment 2c performed a comparison between the top-performing methods established in Experiments 2a and 2b. We used the testing datasets for this experiment. For 16S.B.ALL, 10,000 sequences were chosen by UPP's algorithm for the backbone while the remaining 17,643 sequences were used as the query set. For 16S.3 and 16S.T, 1000 sequences were chosen by UPP as the backbone and the remaining sequences were used as the query set. A larger backbone was chosen for 16S.B.ALL to reflect the larger dataset size compared to 16S.3 and 16S.T. We followed the same procedure as for Experiment 1 to construct the backbone alignments and trees.



(a) Alignment error



(b) Running time

Figure 3: **Experiment 2a: UPP2 against Benchmark Methods on Simulated Fragmentary Datasets** We compare UPP2 to benchmark methods on simulated datasets with fragmentary sequences. All methods except T-COFFEE and MUSCLE were run in their default modes and with 16 threads, when possible. T-COFFEE was run using the regressive mode and MUSCLE was limited to 2 iterations. All datasets have 20 replicates each. The bar indicates the mean while the error bars indicate standard error for alignment error and standard deviation for running time.

4 Results

4.1 Experiment 1: Designing UPP2

In this first experiment we evaluated variants of UPP2, varying (a) use of adjusted or raw bit-scores, (b) using MAGUS or PASTA backbones, (c) changing the value for z (maximum allowed size of subsets before decomposition stops), and (d) use of EarlyStop or Hierarchical as opposed to all-against-all.

On all the datasets we explored (i.e., full-length and also HF versions of 1000M1 and RNASim1000), there were no noteworthy differences in alignment accuracy for any of these modifications, with the exception that using MAGUS instead of PASTA for the backbone alignment improved accuracy (Figure 2 and Supplementary Figures S1 and S2). We also saw that using MAGUS instead of PASTA for the backbone alignment reduced runtime (Supplementary Figure S2) and that using the new search strategies (EarlyStop or Hierarchical) improved runtime even further (Figure 2). Specifically, using UPP2-Hierarchical reduced the runtime by a large margin and UPP2-EarlyStop slightly improved runtime

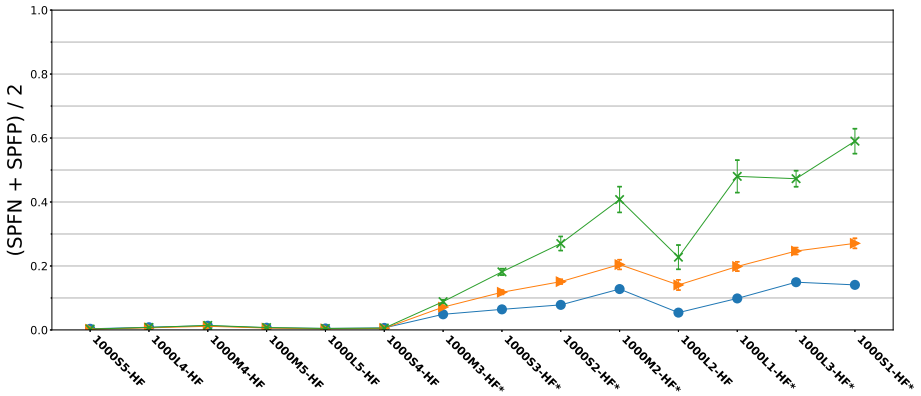


Figure 4: **Experiment 2a: Comparison of UPP2, MAGUS, and PASTA on Simulated Datasets.** We compare UPP2, MAGUS, and PASTA on the ROSE model conditions with fragmentary sequences. Asterisks denote the model conditions on which UPP2 was statistically significantly better than MAGUS. All model conditions have 20 replicates, and each replicate has 1000 sequences. The bar indicates the mean while the error bars indicate standard error.

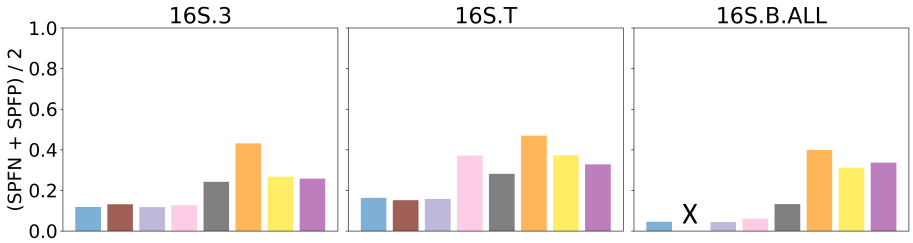
compared to UPP2-Hierarchical.

The runtime improvement obtained through the use of Hierarchical or EarlyStop is not surprising, but the achievement of comparable accuracy was not guaranteed. Although we did not see a difference in accuracy between $z = 2$ compared to $z = 10$, because previous studies (e.g., Mirarab *et al.* (2012)) has suggested the potential for this setting to improve accuracy, we set $z = 2$ for the default for all datasets. Our final default settings for the algorithmic parameters are to use: (a) adjusted bit-scores, (b) z=2, (c) MAGUS for the backbone alignment, and (d) EarlyStop for the search strategy. We denote this variant simply as "UPP2".

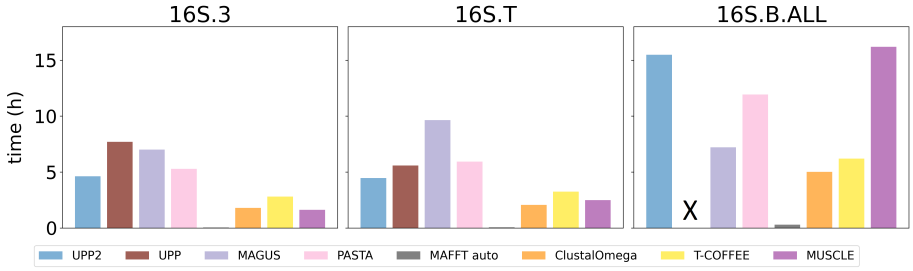
4.2 Experiment 2: UPP2 compared to Benchmark Methods

In this experiment we compare UPP2 to other alignment methods on the testing datasets. Experiment 2(a) explores results on simulated datasets with fragmentary sequences and Experiment 2(b) explores results on biological datasets.

Experiment 2a: Results on simulated datasets with fragmentation. In this experiment we evaluate UPP2 to other alignment methods on simulated datasets with fragmentary sequences (Figure 3a). On these datasets, UPP2 was the most accurate, followed by UPP and MAGUS in that order. PASTA and MAFFT had comparable accuracy to each other and trailed behind the leading group of three methods. Clustal Omega, T-COFFEE, and MUSCLE were the least accurate methods, but Muscle was somewhat more accurate than



(a) alignment error



(b) running time

Figure 5: **Experiment 2b: Comparison of UPP2 to other MSA Methods on Biological Datasets.** The three datasets are from the Comparative Ribosomal Website (Cannone *et al.*, 2002). 16S.3 has 6323 sequences, 16S.T has 7350 sequences, and 16S.B.ALL has 27,643 sequences. MAFFT *auto* mode was used rather than the *linsi* mode due to the large dataset sizes. **X** indicates that UPP could not finish within the time limit (7 days) for the 16S.B.ALL dataset.

the others and Clustal Omega and T-COFFEE tended to perform similar to each other.

The runtime comparison (Figure 3b) shows relative performance is based on the model condition. UPP2, MAGUS, Clustal Omega, and T-COFFEE the fastest methods on three of the six model conditions (1000S4-HF, 1000M3-HF, 1000L5-HF). UPP2, UPP, and Clustal Omega were the fastest methods on the remaining the model conditions (1000S1-HF, 1000M2-HF, and 1000L1-HF). On the first set of model conditions, MAGUS was faster than UPP, and MUSCLE was the slowest. On the second set of model conditions, UPP was faster than MAGUS and MAFFT was the slowest. PASTA was consistently the second slowest method.

Thus, these results establish that UPP2 is at least as accurate as UPP but faster, and that PASTA and MAGUS are the next two most accurate methods. Here we directly compare UPP2, MAGUS, and PASTA to enable a more detailed understanding of their relative accuracy.

Figure 4 shows alignment error for UPP2, MAGUS, and PASTA on the ROSE model conditions (each with fragmentary sequences), sorted (approximately) so that alignment error rates generally increase from left-to-right. While

the three methods are identical on the six easiest model conditions (i.e., the leftmost conditions), as the model condition difficulty increases we see error rates increasing for all methods, but UPP2 error rates increase more slowly than for the others. Thus, across all the more difficult model conditions, we see that UPP2 is much more accurate than MAGUS, which in turn is more accurate than PASTA. Furthermore, accuracy differences between methods UPP2 are statistically significantly on several high fragmentary model conditions, as indicated using asterisks in Figure 4.

An examination of the properties of these model conditions (Supplementary Materials Table S1) shows that these conditions vary significantly in terms of average and maximum p-distances (i.e., normalized Hamming distances), and that these average p-distances generally increase as we move from left to right. Specifically, the first six model conditions (i.e., 1000S4, 1000S5, 1000M4, 1000M5, 1000L4, and 1000L5) range in average p-distances from 49.5% to 50.1%, and the next eight model conditions have increasing average p-distances that range from 66.0% to 69.6%. Thus, increases in average p-distance result in increases in alignment error for all methods, and also increase the gap between methods.

Experiment 2b: Results on biological datasets. Results on the biological datasets (Figure 5(a)) show UPP2 and MAGUS were the most accurate methods, with PASTA being as accurate as the top methods on 16S.B.ALL and 16S.3 but not on 16S.T. UPP was as accurate as UPP2 on 16S.3 and 16S.T but could not finish in the allotted time (seven days) on the 16S.B.ALL dataset. Clustal Omega had the highest alignment error across all biological model conditions. T-COFFEE, MUSCLE, and MAFFT all performed similarly to each other on 16S.3, but MAFFT was able to beat the other two methods on 16S.B.ALL and 16S.T.

Runtime comparisons on the biological datasets (Figure 5(b)) shows differences that depend on the dataset. On the largest of the three biological datasets (16S.B.ALL, with 27,643 sequences), UPP did not finish within the allowed time, and only MAFFT was very fast (finishing in just a few minutes). The remaining methods varied in runtime, with UPP2 and Muscle using the most time (more than 15 hours), followed by MAGUS (more than 7 hours), and then by the remaining methods (between 5 and a little over 6 hours each). On the two smaller datasets (6323 sequences and 7350 sequences), MAFFT was again by far the fastest (finishing in a few minutes). Although runtimes have dropped for the remaining methods on these smaller datasets, some trends can nevertheless be discerned. On the 16S.3 and 16S.T datasets, the remaining methods group into two sets: Clustal Omega, T-Coffee, and Muscle are the fastest (all finishing in at most 4 hours) and UPP2, UPP, MAGUS, and PASTA are the slowest (finishing in 4 to 10 hours).

Thus, although UPP tied for most accurate when it could complete, it was too slow to complete on the largest biological dataset. UPP2 and MAGUS reliably had good accuracy (tieing for best) and completing within reasonable times. The comparison between UPP2 and MAGUS shows indistinguishable accuracy on these datasets, with UPP2 faster than MAGUS on two of the three datasets.

5 Discussion

In our first experiment we selected UPP+adj+EarlyStop with MAGUS backbones as the most promising variant of UPP2, since it provided improved accuracy over UPP (as a result of using the MAGUS backbone) and improved runtime (as a result of the EarlyStop search strategy). This variant, which we more simply refer to as "UPP2", overall provides advantages over other alignment methods. Specifically, on datasets with fragmentation, it provides better accuracy than all other methods; the only method that comes close for accuracy is UPP, but the change in search strategy (EarlyStop vs all-against-all) makes UPP2 much faster. On simulated datasets without any fragmentation (Supplementary Materials, Figure S3), UPP2 is a very good method but not the most accurate (instead, MAGUS is the most accurate and UPP2 is in second place);; however, given the levels of sequence length heterogeneity seen in biological datasets (see Figure 1 in Nguyen *et al.* (2015)), results on simulated datasets without sequence length heterogeneity are not as relevant for understanding performance on biological datasets.

We also see that the rate of evolution impacts absolute and relative accuracy of alignment methods. On datasets with low rates of evolution all methods can be highly accurate even if there is a high level of fragmentation; however, differences appear under higher rates, where UPP2 dominates all the other methods, and the gap between UPP2 and the other methods increases with the rate of evolution (Figure 4). This trend explains in part why UPP2 matches but does not consistently improve on UPP (other than for runtime) nor on MAGUS or PASTA on the CRW datasets (Figure 5a), which have low evolutionary diameters (as evidenced by the low p-distances in these datasets).

UPP2 is also generally reasonably fast, and generally at least as fast as the other most accurate methods (UPP, MAGUS, and PASTA). However, UPP2 is not nearly as fast as MAFFT –auto (which is the fastest of the methods we explored) and was slow on the 16S.B.ALL dataset. The biggest component of the runtime for UPP2 on 16S.B.ALL is Stage 3, which took about 9 hours to place 17,643 sequences into its backbone alignment of 10,000 sequences; the second most expensive step is Stage 1 (a bit more than 5 hrs), computing the backbone alignment using MAGUS. Since the runtime for Stage 3 is linear in the number of query sequences, to reduce the runtime the backbone alignment could have been bigger. On this particular dataset, MAGUS provided good accuracy and was fast, suggesting that a larger backbone (computed by MAGUS) would have been a safe substitution. Exploring the impact of changing the backbone size on accuracy and runtime is a direction for future research.

6 Conclusions

Given the fundamental nature of multiple sequence alignment in many bioinformatics pipelines and the availability of low cost DNA sequencing, the estimation of multiple sequence alignments on large datasets is a common step in

much biological discovery. However, many modern biological datasets exhibit substantial sequence length heterogeneity, and only a few methods have been able to provide good accuracy under these conditions. The previous most accurate method for aligning datasets with fragmentary sequences was UPP, but UPP's all-against-all approach made it computationally intensive. By replacing this search strategy with the EarlyStop approach, UPP2 achieves the same high accuracy but is much faster than UPP. The improvement in runtime without degradation of accuracy provided by UPP2 is encouraging, and suggests the potential for even more significant advances.

References

Blackburne, B. P. and Whelan, S. (2013). Class of multiple sequence alignment algorithm affects genomic analysis. *Molecular Biology and Evolution*, **30**(3), 642–653.

Bode, B. *et al.* (2013). The Blue Waters Super-System for Super-Science. In *Contemporary High Performance Computing*, Chapman & Hall/CRC Computational Science, pages 339–366. Chapman and Hall/CRC.

Cannone, J. J. *et al.* (2002). 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), 1–31.

Durbin, R. *et al.* (1998). *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press.

Eddy, S. R. (2011). Accelerated Profile HMM searches. *PLoS Computational Biology*, **7**(10), e1002195.

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, **32**(5), 1792–1797.

Garriga, E. *et al.* (2019). Large multiple sequence alignments with a root-to-leaf regressive method. *Nature Biotechnology*, **37**(12), 1466–1470.

Guo, S. *et al.* (2009). Large-scale simulation of RNA macroevolution by an energy-dependent fitness model. arXiv:0912.2326.

Katoh, K. and Standley, D. M. (2013). MAFFT multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, **30**(4), 772–780.

Liu, K. *et al.* (2009). Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. *Science*, **324**(5934), 1561–1564.

Liu, K. *et al.* (2011). SATé-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. *Systematic Biology*, **61**(1), 90–90.

Löytynoja, A. and Goldman, N. (2005). An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences*, **102**(30), 10557–10562.

Mirarab, S. (2017). [Dataset] 16S cleaned alignment. Google Sites - UCSD Engineering. `https://sites.google.com/eng.ucsd.edu/datasets/alignment/16s23s` Accessed: 2021-12-16.

Mirarab, S. (2022). Github page for SEPP, TIPP, and UPP. https://github.com/smirarab/sepp, last accessed Feb 25, 2022.

Mirarab, S. and Warnow, T. (2011). FastSP: linear time calculation of alignment accuracy. *Bioinformatics*, **27**(23), 3250–3258.

Mirarab, S. *et al.* (2012). SEPP: SATé-Enabled Phylogenetic Placement. In *Biocomputing 2012*, pages 247–258. World Scientific.

Mirarab, S. *et al.* (2015). PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences. *Journal of Computational Biology*, **22**(5), 377–386.

Nguyen, N.-p. D. *et al.* (2015). Ultra-large alignments using phylogeny-aware profiles. *Genome Biology*, **16**(1), 1–15.

Rao, R. M. *et al.* (2021). MSA Transformer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8844–8856. PMLR.

Shen, C. *et al.* (2021). MAGUS+eHMMs: improved multiple sequence alignment accuracy for fragmentary sequences. *Bioinformatics*, **38**(4), 918–924.

Shen, C. *et al.* (2022). WITCH: improved multiple sequence alignment through weighted consensus HMM alignment. Under review.

Sievers, F. *et al.* (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, **7**(1), 539.

Smirnov, V. (2021). Recursive MAGUS: Scalable and accurate multiple sequence alignment. *PLoS Computational Biology*, **17**(10), e1008950.

Smirnov, V. and Warnow, T. (2021a). [Dataset] Phylogeny Estimation Given Sequence Length Heterogeneity. Dryad. `https://doi.org/10.5061/dryad.95x69p8h8`.

Smirnov, V. and Warnow, T. (2021b). MAGUS: Multiple sequence Alignment using Graph clUStering. *Bioinformatics*, **37**(12), 1666–1672.

Smirnov, V. and Warnow, T. (2021c). Phylogeny estimation given sequence length heterogeneity. *Systematic Biology*, **70**(2), 268–282.

Suchard, M. A. and Redelings, B. D. (2006). BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics*, **22**(16), 2047–2048.

Söding, J. (2004). Protein homology detection by HMM–HMM comparison. *Bioinformatics*, **21**(7), 951–960.

Supplementary Materials

Minhyuk Park[1] Stefan Ivanovic[1] Gillian Chu[1]
Chengze Shen Tandy Warnow[1*]

[1]Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820
[*]To whom correspondence should be addressed.

Contents

List of Figures

List of Tables

S1 Method Versions and Commands

FastSP

- Version: 1.7.1
- Availability: `https://github.com/smirarab/FastSP`
- Note: -ml and -mlr flags are omitted for MAFFT alignments since MAFFT only outputs lowercase characters
- Command:

```
java -jar FastSP.jar -ml -mlr -r <REFERENCE ALIGNMENT> -e <ESTIMATED ALIGNMENT>
```

MAFFT

- Version: 7.487
- Availability: `https://mafft.cbrc.jp/alignment/software/`
- Command:

```
linsi --thread 16 <SEQUENCE FILE> 1> <OUTPUT>/mafft.fasta
```

MUSCLE

- Version: 3.8.31
- Availability: `https://www.drive5.com/muscle/`
- Command:

```
muscle -in <SEQUENCE FILE> -out <OUTPUT>/muscle.fasta
```

ClustalOmega

- Version: 1.2.4
- Availability: `http://www.clustal.org/omega/`
- Command:

```
clustalo --threads=16 --in <SEQUENCE FILE> --out <OUTPUT>/clustalo.fasta
```

MAGUS

- Commit id: 95522ec9539575189a0a2f90baaf81cbde480034
- Availability: `https://github.com/vlasmirnov/MAGUS`
- Command:

```
python MAGUS/magus.py -d <OUTPUT> -i <SEQUENCE FILE> -o <OUTPUT>/magus.fasta
```

T-COFFEE

- Version: Version 13.45.0.4846264
- Availability: `http://www.tcoffee.org/`
- Command:

```
t_coffee -thread=16 -reg -seq <SEQUENCE FILE> -outfile <OUTPUT>/t_coffee.fasta
```

PASTA

- Version: PASTA v1.9.0
- Availability: `https://github.com/smirarab/PASTA`
- Command:

```
python run_pasta.py -i <SEQUENCE FILE> --num-cpus 16 -o <OUTPUT> --temporaries <OUTPUT>
```

UPP

- Version: 4.5.1
- Availability: `https://github.com/smirarab/sepp`
- Note: When letting UPP choose the backbone sequneces, the alignment and tree flags can be omitted.
- Command:

```
python run_upp.py -c <CONFIG FILE>
```

UPP Config

```
[commandline]
sequence_file=<SEQUENCES>
backboneSize=<BACKBONE SIZE>
alignmentSize=<DECOMPOSITION SIZE>
molecule=<dna/rna/amino>
cpu=16
tempdir=<TEMP DIRECTORY>
outdir=<OUTPUT DIRECTORY>
```

UPP2

- Commit ID: 0ba1b31829c1982d8ffbfe94452b7174663d13f3
- Availability: `https://github.com/gillichu/sepp`
- Note: When letting UPP2 choose the backbone sequneces, the alignment and tree flags can be omitted.
- Command:

```
python run_upp.py -c <CONFIG FILE>
```

UPP2 Config

```
[commandline]
sequence_file=<QUERY SEQUENCES>
alignment=<BACKBONE FASTA>
tree=<BACKBONE TREE>
backboneSize=<BACKBONE SIZE>
alignmentSize=<DECOMPOSITION SIZE>
molecule=<dna/rna/amino>
cpu=16
tempdir=<TEMP DIRECTORY>
outdir=<OUTPUT DIRECTORY>

[upp2]
decomp_only=True
bitscore_adjust=True
hier_upp=False
early_stop=False
```

UPP2-Hierarchical Config

```
[commandline]
sequence_file=<QUERY SEQUENCES>
alignment=<BACKBONE FASTA>
tree=<BACKBONE TREE>
backboneSize=<BACKBONE SIZE>
alignmentSize=<DECOMPOSITION SIZE>
molecule=<dna/rna/amino>
cpu=16
tempdir=<TEMP DIRECTORY>
outdir=<OUTPUT DIRECTORY>

[upp2]
decomp_only=True
bitscore_adjust=True
hier_upp=True
early_stop=False
```

UPP2-EarlyStop Config

```
[commandline]
sequence_file=<QUERY SEQUENCES>
alignment=<BACKBONE FASTA>
tree=<BACKBONE TREE>
backboneSize=<BACKBONE SIZE>
alignmentSize=<DECOMPOSITION SIZE>
molecule=<dna/rna/amino>
cpu=16
tempdir=<TEMP DIRECTORY>
outdir=<OUTPUT DIRECTORY>

[upp2]
decomp_only=True
bitscore_adjust=True
hier_upp=True
early_stop=True
```

S2 Dataset properties

Table S1: Dataset properties. We show the average and maximum p-distances (normalized Hamming distances) and number of sequences in each of the study datasets. Most of these datasets are studied in three versions: unmodified (i.e., without fragmentation), low-fragmentation (LF, where 25% of the sequences are shortened to 50% of their length) and high-fragmentation (HF, where 50% of the sequences are shortened to 25% of their length).

Name	Sim/Bio	# Sequences	avg. p-dist.	max. p-dist.
1000S1	Sim	1000	0.694	0.768
1000S2	Sim	1000	0.693	0.768
1000S3	Sim	1000	0.686	0.763
1000S4	Sim	1000	0.501	0.608
1000S5	Sim	1000	0.498	0.611
1000M1	Sim	1000	0.695	0.769
1000M2	Sim	1000	0.684	0.762
1000M3	Sim	1000	0.660	0.741
1000M4	Sim	1000	0.495	0.606
1000M5	Sim	1000	0.499	0.602
1000L1	Sim	1000	0.695	0.769
1000L2	Sim	1000	0.696	0.769
1000L3	Sim	1000	0.687	0.763
1000L4	Sim	1000	0.500	0.608
1000L5	Sim	1000	0.496	0.606
RNASim1000	Sim	1000	0.411	0.609
16S.3	Bio	6323	0.315	0.833
16S.T	Bio	7350	0.345	0.901
16S.B.ALL	Bio	27643	0.210	0.769

S3 Extra figures

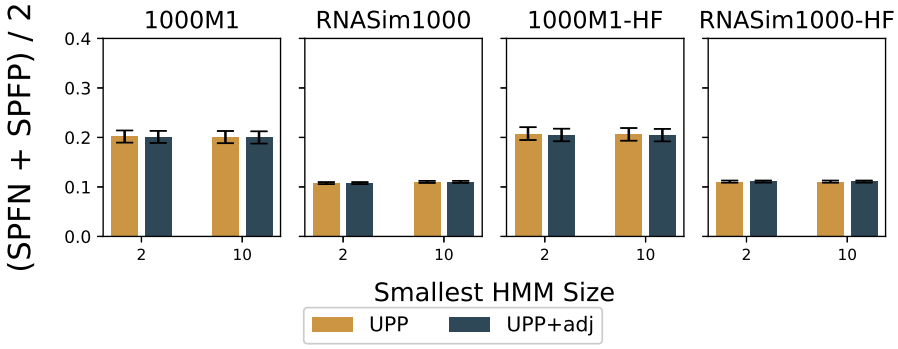


Figure S1: **Experiment 1: Impact of Adjusted Bit-score and Stopping Rule (size of smallest subsets) on Alignment Error** UPP uses the raw bit-scores while UPP+adj uses the adjusted bit-scores. Both methods perform an all-against-all search of HMMs to query sequences. Each subfigure shows two values for z, the size of the smallest subset within the decomposition strategy (i.e., stopping rule). The bar indicates the mean while the error bars indicate standard error.

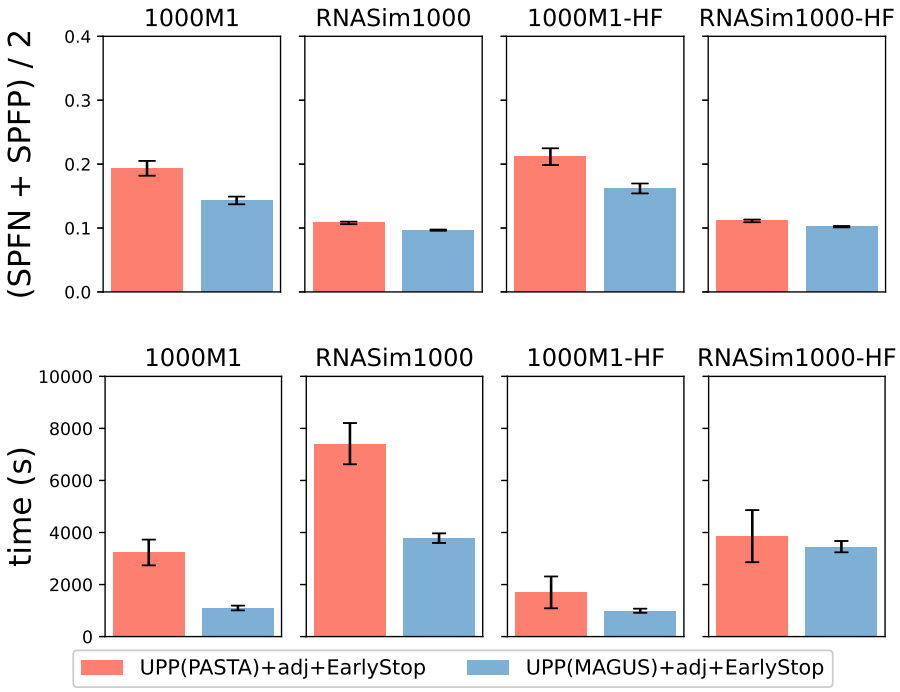
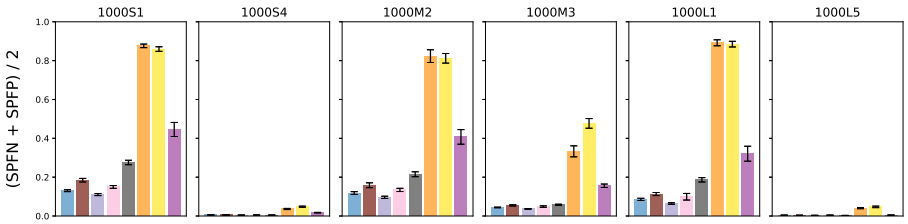
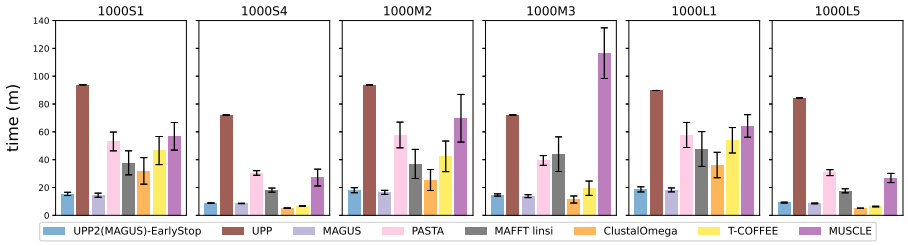


Figure S2: **Experiment 1** Impact of choice of backbone alignment method (MAGUS vs PASTA) and Stopping Rule (size of smallest subsets) on alignment accuracy and total runtime. 1000M1 has 19 replicates, RNASim1000 has 20 replicates, 1000M1-HF has 19 replicates, and RNASim1000-HF has 20 replicates. The means are shown with error bars indicating standard error for alignment error and standard deviation for running time.



(a) alignment error



(b) running time

Figure S3: **Experiment 2: UPP2 compared to benchmark alignment methods on simulated datasets without fragmentation.** We show alignment error and runtime of UPP2 (i.e., UPP+adj+EarlyStop with MAGUS backbones) compared to other alignment methods. All methods except T-COFFEE and MUSCLE were run in their default modes and with 16 threads, when possible. T-COFFEE was run using the regressive mode and MUSCLE was limited to 2 iterations. All datasets have 20 replicates each. The means are shown with error bars indicating standard error for alignment error and standard deviation for running time.

S4 Results on simulated datasets without fragmentary sequences

Figure S3 compares methods on simulated datasets without any introduced fragmentation, showing both alignment error and running time. On these datasets, UPP2, UPP, MAGUS, and PASTA were the four leading methods across the simulated model conditions. Within this leading group of methods, MAGUS was the most accurate alignment method, closely followed by UPP2, PASTA, and UPP, in that order. Clustal Omega and T-COFFEE were the least accurate methods. MUSCLE, although more accurate than Clustal Omega and T-COFFEE, was less accurate than the leading group of four methods. UPP2 and MAGUS were the fastest methods, followed by Clustal-Omega and T-COFFEE. UPP was almost always the slowest, but PASTA, Muscle, and MAFFT linsi were typically also slow. Thus, if selecting from the most accurate methods, UPP2 and MAGUS are the only ones that provide competitive accuracy and also fast running times.

S5 Bitscore to Probability When HMMs are Different Sizes

The bitscore of a query sequence given a HMMER HMM is $\log_2 \frac{P(q|H)}{P(q|H_0)}$ where H is the HMM, q is the query sequence, and H_0 is the null model, or the random model.

Using Bayes' theroem, we arrive at the probability of H_i generating sequence q as follows.

$$\begin{aligned} P(H_i|q) &= \frac{P(q|H_i) \cdot P(H_i)}{P(q)} \\ &= \frac{P(q|H_i) \cdot P(H_i)}{\Sigma_{j=1}^{n} P(q|H_j) \cdot P(H_j)} \end{aligned}$$

where n is the number of HMMs $(H_i ... H_n)$.

If we assume that the more sequences the HMM is trained on, the more likely the HMM is to output a sequence, then we can transform the above into the following.

$$\begin{aligned} P(H_i|q) &= \frac{P(q|H_i) \cdot \frac{s_i}{S}}{\Sigma_{j=1}^{n} P(q|H_j) \cdot \frac{s_j}{S}} \\ &= \frac{1}{\Sigma_{j=1}^{n} \frac{P(q|H_j) \cdot s_j}{P(q|H_i) \cdot s_i}} \\ &= \frac{1}{\Sigma_{j=1}^{n} 2^{\log_2 \frac{P(q|H_j) \cdot s_j}{P(q|H_i) \cdot s_i}}} \end{aligned}$$

where s_i is the number of sequences that HMM H_i was trained on and S is the total number of sequences that the HMMs were trained on.

From the definition of Bitscores, we can derive the following.

$$\begin{aligned} BS(H_j) - BS(H_i) &= \log_2 \frac{P(q|H_j)}{P(q|H_0)} - \log_2 \frac{P(q|H_i)}{P(q|H_0)} \\ &= \log_2 \frac{P(q|H_j)}{P(q|H_i)} \end{aligned}$$

So

$$\begin{aligned} P(H_i|q) &= \frac{1}{\Sigma_{j=1}^{n} 2^{\log_2 \frac{P(q|H_j) \cdot s_j}{P(q|H_i) \cdot s_i}}} \\ &= \frac{1}{\Sigma_{j=1}^{n} 2^{BS(H_j) - BS(H_i) + \log_2 \frac{s_j}{s_i}}} \end{aligned}$$