PASTA: Ultra-Large Multiple Sequence Alignment

Siavash Mirarab, Nam Nguyen, and Tandy Warnow

University of Texas at Austin - Department of Computer Science
2317 Speedway, Stop D9500 Austin, TX 78712
{smirarab,namphuon,tandy}@cs.utexas.edu

Abstract. In this paper, we introduce a new and highly scalable algorithm, PASTA, for large-scale multiple sequence alignment estimation. PASTA uses a new technique to produce an alignment given a guide tree that enables it to be both highly scalable and very accurate. We present a study on biological and simulated data with up to 200,000 sequences, showing that PASTA produces highly accurate alignments, improving on the accuracy of the leading alignment methods on large datasets, and is able to analyze much larger datasets than the current methods. We also show that trees estimated on PASTA alignments are highly accurate – slightly better than SATé trees, but with substantial improvements relative to other methods. Finally, PASTA is very fast, highly parallelizable, and requires relatively little memory.

Keywords: Multiple sequence alignment, Ultra-large, SATé.

1 Introduction and Motivation

Multiple sequence alignment (MSA) is a basic step in many bioinformatics analyses, including predicting the structure and function of RNAs and proteins and estimating phylogenies. Yet only a handful of the many MSA methods are able to analyze large datasets with 10,000 or more sequences. Performance studies evaluating MSA methods on large datasets have shown that some MSA methods can produce highly accurate alignments, as measured by traditional alignment criteria (sum-of-pairs or column scores) for sufficiently slowly evolving sequence datasets (e.g., [1]). However, other studies, focusing on phylogeny estimation from nucleotide datasets, have found that only a handful of MSA methods can provide good enough alignments on nucleotide datasets with 10,000 or more sequences to produce highly accurate trees, even when trees are estimated using the best maximum likelihood heuristics [2–5]. However, these studies have relied upon benchmarks with at most 28,000 sequences, and so little is known about alignment accuracy and its impact on tree accuracy for larger datasets. Yet, phylogenetic analyses of sequence datasets containing more than 100,000 nucleotide sequences are being attempted by at least two groups that we are aware of: the iPTOL project [6] and the Thousand Transcriptome project (1KP)[7].

R. Sharan (Ed.): RECOMB 2014, LNBI 8394, pp. 177–191, 2014.

In this paper we present PASTA, "Practical Alignments using SATé and TrAnsitivity", a new method for ultra-large multiple sequence alignment of nucleotide sequence datasets. PASTA begins with an alignment and tree estimated using a very simple profile HMM-based technique and then re-aligns the sequences using the tree. If desired, a new tree can be estimated on the new alignment, and the algorithm can iterate.

The key to the accuracy and scalability of PASTA is the novel technique it uses for estimating an alignment on a guide tree. As in SATé [3], PASTA uses the centroid edge dataset decomposition technique and computes MAFFT -linsi [8] alignments on the subsets; however, PASTA and SATé merge these subset alignments into an alignment on the full dataset using very different techniques. While SATé uses Opal [9] (or Muscle [10, 11], if the dataset is too large) to hierarchically merge all the subset alignments into a single alignment, PASTA uses Opal only to merge pairs of adjacent subset alignments, producing overlapping alignments, then treats each resultant alignment as an equivalence relation and uses transitivity to merge these larger alignments. The result is a very fast re-alignment method that is highly parallelizable and easily scales to large datasets. Furthermore, this re-alignment step in PASTA is a negligible fraction of the PASTA analysis, whereas the re-alignment step in SATé is the majority of its running time on large datasets (44% of the running time for datasets with 10K sequences, and 78% of the time for datasets with 50K sequences). Thus, PASTA is dramatically faster than SATé on large datasets. Interestingly, PASTA produces more accurate alignments and trees than SATé. We demonstrate PASTA's speed and accuracy on a collection of datasets, including a 200K-sequence RNASim dataset [12], which we align in less than 24 hours using PASTA on a 12-core machine.

2 PASTA

We describe PASTA and present some theorems about its performance guarantees and running time; due to space limits, proofs are provided in the appendix.

PASTA uses an iterative divide-and-conquer strategy to align an input set S of sequences, and uses the following input parameters: a starting tree (default our HMM-based profile alignment technique, described below), subset size k (default 200), a subset alignment technique (default MAFFT -linsi), an alignment merger technique (default OPAL), and a stopping rule (default 3 iterations).

The first iteration begins with the starting tree, and subsequent iterations begin with the tree estimated in the previous iteration. Each iteration involves six steps, shown in Figure 1.

Starting Tree: PASTA can begin with any reasonable starting tree, but here we describe the simple technique, similar to that used in [14, 15], that we use in these experiments. We take a random subset X of 100 sequences from S and compute a SATé alignment A on the set; this is called the "backbone alignment". We then use HMMER [16, 17] to compute a Hidden Markov Model on A, and to align all sequences in $S-X$ one by one (and independently) to A, and hence build

Step 1 Decompose the input set S into subsets $S_1 \ldots S_m$ of size at most k.
Step 2 Compute a spanning tree T^* to connect the subsets $S_1 \ldots S_m$.
Step 3 Align each subset using the subset alignment technique.
Step 4 Merge the two alignments on endpoints of each edge in T^*.
Step 5 Use successive applications of transitive closure to merge the overlapping and compatible alignments obtained in Step 4.
Step 6 Compute a maximum likelihood (ML) tree on the full MSA using FastTree-2 [13].

Fig. 1. Algorithmic steps of PASTA for each iteration

an alignment of the full dataset. We then construct a ML tree on this alignment using FastTree-2 [13]. If this technique fails to produce an alignment on the full set of sequences (which can happen if HMMER considers some sequences unalignable), we randomly add the unaligned sequences into the tree obtained on the alignment obtained by HMMER.

Step 1: We use the centroid decomposition technique in SATé [3] on the current guide tree to divide the sequence set into disjoint sets, $S_1, \ldots, S_m$, each with at most k sequences. If the tree has at most k leaves, we return the set of sequences; otherwise, we find an edge in the tree that splits the set of leaves into roughly equal sizes, remove it from the tree, and then recurse on each subtree.

Step 2: We compute a spanning tree T^* on the subsets, $S_1, S_2, \ldots, S_m$, as follows. For every i, we compute the set of nodes v in the guide tree that are on a path between two leaves that both belong to S_i, and we label all these nodes by S_i; thus, if v is a leaf and belongs to S_i, we label v by S_i. Then, if some nodes are not yet labelled, we propagate labels from nodes to unlabelled neighbors (breaking ties by using the closest neighbor according to branch lengths in the guide tree) until all nodes are labelled. We then collapse edges that have the same label at the endpoints. The result is a spanning tree on $S_1, S_2, \ldots, S_m$.

Step 3: We compute MSAs on each S_i using the subset alignment method specified by the user. We refer to each such alignment as a "Type 1 sub-alignment".

Step 4: Every node in T^* is labelled by an alignment subset for which we have a Type 1 sub-alignment from Step 3. For every edge (v, w) in T^*, we use the specified alignment merger technique to merge the Type 1 sub-alignments at v and w; this produces a new set of alignments, each containing at most $2k$ sequences, which are called "Type 2 sub-alignments". We require that the merger technique used to compute Type 2 sub-alignments not change the alignments on the Type 1 sub-alignments; therefore, Type 2 sub-alignments induce the Type 1 sub-alignments computed in Step 2.

Step 5*:* We compute the transitivity merge through a sequence of pairwise transitivity mergers. To motivate this technique, we note that every MSA defines an equivalence relation on the letters within its sequences, whereby two letters are in the same equivalence class *if and only if* they are in the same column [18]. Hence given two alignments A and B that induce identical alignments on their shared sequences (called *overlapping compatible* alignments henceforth), we can define an equivalence relation on the union of the letters from their sequence subsets, as follows: a and b are in the same equivalence class for the merged alignment if and only if at least one of the following is true: (1) they are in the same equivalence class in A or B, or (2) there is some letter c such that a and c are in the same equivalence class in one alignment, and b and c are in the same equivalence class for the other alignment. This is the basis for the pairwise transitivity merger, but the spanning tree enables this technique to extend to a set of alignments through a sequence of pairwise transitivity mergers in a computationally efficient manner. We provide details for the transitivity merge in the appendix.

Step 6: If an additional iteration (or a tree on the alignment) is desired, we run FastTree-2 [13] to estimate a maximum likelihood tree on the MSA produced in the previous step. To speed up this step, we mask all sites that have more than 99.9% gaps in the alignment obtained in Step 5. Note that PASTA's alignment merging step is conservative in introducing new homologies (only what is necessary through transitivity is added) and thus PASTA tends to produce many gappy columns. Masking these highly gapped columns is harmless for the tree estimation step but has a dramatic effect on the running time.

Running Time Considerations. PASTA keeps alignments in the memory in a condensed format by representing each sequence by its unaligned sequence and column indices for each letter. So, for example, `('ACCA',[1,3,5,6])` corresponds to `'A-C-CA'`. This format reduces the memory requirement of PASTA, as well as the running time for each transitivity merge.

As long as each pairwise transitivity merge is performed correctly, the final output multiple sequence alignment does not depend on the order in which edges of the spanning tree are processed, but the order can impact the running time. However, if we merge sub-alignments using the reverse order of the centroid edge deletions, then the running time can be bounded, as follows:

Theorem 1. *Given m Type 1 alignments and $m-1$ Type 2 alignments, the algorithm to compute the transitivity merge of these alignments uses $O(Km \log m + Lm)$ time, where K is the maximum length of any sequence (not counting gaps) in any Type 1 alignment, and L is the length of output alignment.*

The bound given in the theorem is achievable using an order of edge contractions that reverses the order of centroid edge deletions; however, an *arbitrary* order of edge contractions can result in a worst case $O(Km^2 + Lm)$ running time; see Supplementary Material [19] for discussion and proof.

Table 1. Empirical statistics of large datasets based on reference alignments. Number of sequences, number of sites, proportion of gap characters, maximum p-distance, and average p-distance are given for each dataset (the p-distance is the fraction of sites in which two sequences differ). For 10K RNASim datasets, the given values are averages over 10 replicates. For RNASim datasets with 100K and 200K sequences, pairwise distances could not be computed; however, since 100K and 200K datasets are random subsamples from the same alignment as 10K and 50K datasets, their alignment statistics are likely very close to those of 10K and 50K.

	Dataset	# Sequences	# Sites	Proportion gaps	Max p-dist	Avg p-dist
Gutell	16S.B.ALL	27,643	6,857	0.800	0.769	0.210
Gutell	16S.T	7,350	11,856	0.874	0.900	0.345
Gutell	16S.3	6,323	8,716	0.821	0.832	0.315
RNASim	10,000	10,000	8,637	0.820	0.616	0.410
RNASim	50,000	50,000	12,400	0.875	0.620	0.410
RNASim	100,000	100,000	14,316	0.891	≈ 0.62	≈ 0.410
RNASim	200,000	200,000	16,365	0.905	≈ 0.62	≈ 0.410

3 Experimental Setup

Datasets. To evaluate the accuracy on moderate size datasets, we use 1000-taxon simulated datasets from [2]. For evaluating performance on larger datasets, we used RNASim, a simulated RNA dataset with 1,000,000 sequences [12], subsampling it to create datasets with 10,000, 50,000, 100,000, and 200,000 sequences. For the 10K case we created 10 different replicates, but for other cases, due to running time requirements, we created only one replicate. Finally, we use three large 16S biological datasets obtained from the Gutell lab [20], and previously studied in [2]. These datasets include 16S.3 with 6323 sequences, 16S.T with 7350 sequences, and 16S.B.ALL with 27,643 sequences. The reference alignments for the biological datasets are based on secondary structure, and the reference trees are computed on these reference alignments using RAxML [21], with all edges having bootstrap support less than 75% contracted; using other thresholds produces similar results (see online supplementary materials [19]). The reference alignments and trees for the simulated datasets are the true alignment and true trees, which are known because they are the result of a simulation process. Table 1 shows more statistics about the reference alignments.

Methods. We compare PASTA to SATé, Muscle, MAFFT-Profile [22], ClustalW (quicktree algorithm) [23], and also to our approach for obtaining the starting alignment and tree. PASTA results are based on the default settings: three iterations, subset size set to 200, MAFFT-linsi used on subsets, and Opal used for merging alignments. The starting tree is obtained by the technique described in Section 2. MAFFT-Profile is a version of MAFFT that can add new sequences into an existing backbone alignment [22]; we provide MAFFT-Profile the same backbone alignment that we use for the starting tree of PASTA. We run SATé

with identical starting trees as PASTA, and we also run SATé for three iterations. Due to high computational costs of OPAL on large datasets, we use Muscle for merging alignments inside SATé for datasets with 5,000 sequences or more, and otherwise we use the default settings in SATé. Finally, we use FastTree-2 to compute ML trees on each alignment. See supplementary material [19] for commands and version numbers.

Criteria. We measure the alignment accuracy, tree error, and running time. Alignment accuracy is measured using FastSP [18] with two different metrics: the SP-score (the percentage of homologies in the reference alignment recovered in the estimated alignment) and the modeler score (the percentage of homologies in the estimated alignment that are correct), averaged together to get one measure. Note that SP-score is the complement of the SP-FN error rate, and the modeler-score is the complement of the SP-FP error rate; thus our measure of alignment accuracy is influenced equally by false positive and false negative homologies. In addition, we also report the number of columns that are recovered entirely correctly in the estimated alignment (TC score). The standard error metric for tree estimation is the bipartition distance, also known as the Robinson-Foulds (RF) rate; however, this metric is not appropriate when the reference tree is not fully resolved, as is the case for the biological datasets. Therefore, we use the False Negative (FN) rate, which is the percentage of true tree edges missing in the estimated trees, to evaluate estimated trees. Note that the FN rate is identical to the RF rate when both estimated and reference trees are binary.

Computational Platform. We ran all methods on the Lonestar Linux cluster at TACC [24], and each run was given one node with 12 cores and 24 GB of memory. Since running time on Lonestar is limited to 24 hours, we were only able to run techniques that could finish in 24 hours (see below). However, PASTA and SATé are iterative techniques, and we allowed them to perform as many iterations (but no more than three) as they could complete within 24 hours. We report the wall clock time in all cases.

4 Results

Ability to complete analyses. We report which methods completed analyses within 24 hrs using 12 cores and 24 GB of memory. All methods completed on all datasets with at most 10,000 sequences. On 16S.B.ALL, all methods except for ClustalW finished. However, ClustalW, Muscle, and SATé-2 failed to complete on the RNASim datasets with 50,000 sequences or more, and MAFFT-profile failed to complete on the RNASim dataset with 200K sequences. On 100k RNAsim, PASTA finished 2 iterations in 24 hours, and on 200k, PASTA was able to complete one iteration and was the only method that could run.

1000-Taxon datasets from SATé papers. We studied PASTA on the 1000-taxon simulated datasets from [2, 3] and observed that PASTA trees matched the

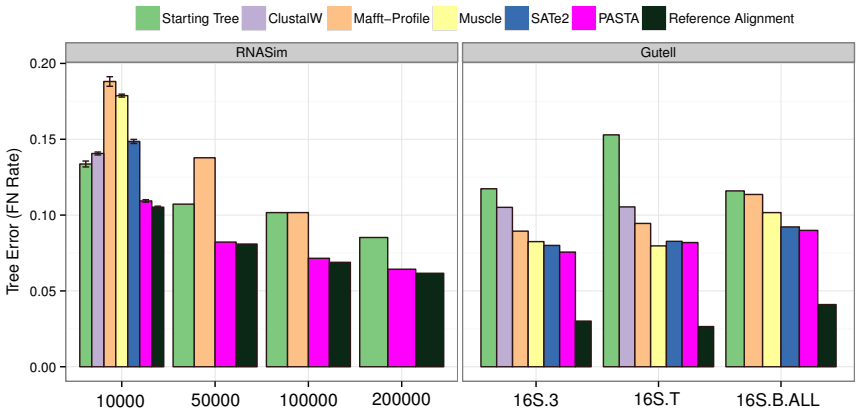

Fig. 2. Tree error rates on RNASim 10K-200K (left) and biological (right) datasets. We show missing branch rates for maximum likelihood trees estimated on the reference alignment as well as alignments computed using PASTA and other methods; results not shown indicate failure to complete within 24 hours using 12 cores on the datasets. Error bars on 10,000 RNASim dataset show standard error over 10 replicates.

accuracy of SATé trees and had improved alignment accuracy; PASTA was also more accurate than the other methods we tested (Opal, Muscle, and MAFFT); see supplementary materials for these results [19].

Tree Error on RNASim and biological datasets. Figure 2 reports results for tree error rates of ML trees on the reference alignments and on the different estimated alignments for the RNASim and biological datasets. On the RNASim data, PASTA returns the most accurate trees, coming very close to FastTree trees on the reference (true) alignment. The difference between PASTA and trees on the next most accurate alignment is very large. Note also that only PASTA and its starting tree complete within the time limit on the 100K and 200K sequence datasets. Furthermore, PASTA has very low error rates overall (e.g., only 6.4% tree error on the 200K dataset).

We also show results for the biological datasets using the 75%-support reference trees. FastTree-2 trees computed on the reference alignments had the best accuracy. PASTA, SATé, and Muscle came next, and the remaining methods had poorer accuracy.

Alignment Accuracy on the RNASim and biological datasets. Figure 3 compares methods with respect to two ways of evaluating accuracy - the total column score (TC) and the average of the SP-score and modeler score. On the RNASim data, PASTA returns by far the most accurate alignments of all methods tested according to TC, and its SP-scores are better than all other methods except the starting alignment. Furthermore, the PASTA alignment had high accuracy: on the 200K dataset, its sum-of-pairs accuracy was 88% and more than 800 columns

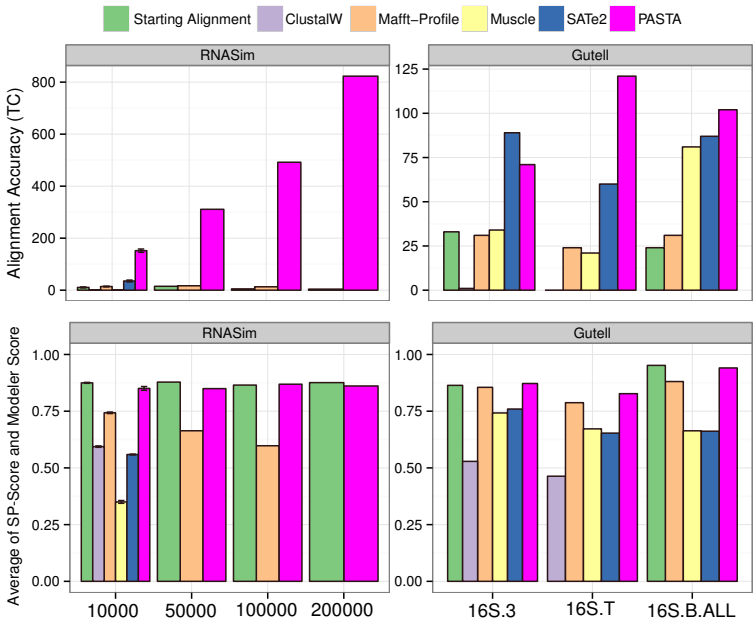

Fig. 3. Alignment accuracy on the RNASim 10K-200K (left) and biological (right) datasets. We show the number of correctly aligned sites (top) and the average of the SP-score and modeler score (bottom). The starting alignment was incomplete on the 16S.T dataset, and so no result is shown for the starting alignment on that dataset.

were recovered entirely correctly. Another interesting trend is that as the number of sequences increases, the alignment accuracy decreased for MAFFT-profile but not for PASTA.

The biological datasets are smaller (see Table 1) and so are not as challenging. On the 16S.T dataset, the starting alignment did not return an alignment with all the sequences on the 16S.T dataset because HMMER considered one of the sequences unalignable. However, the starting alignment technique had good SP-scores for the other two datasets. Of the remaining methods, PASTA has the best sum-of-pairs scores (bottom panel), and MAFFT-profile has only slightly poorer scores; the other methods are substantially poorer. With respect to TC scores, on 16S.B.ALL and 16S.T, PASTA is in first place and SATé is in second place, but they swap positions on 16S.3. TC scores for the other methods are clearly less accurate, though Muscle does fairly well on the 16S.B.ALL dataset.

Comparison to SATé on 50,000 taxon dataset. SATé could not finish even one iteration on the RNASim with 50,000 sequences running for 24 hours and given 12 CPUs on TACC. However, we were able to run two iterations of SATé on a separate machine with no running time limits (12 Quad-Core AMD Opteron(tm) processors, 256GB of RAM memory). Given 12 CPUs, each iteration of SATé takes roughly 70 hours, compared to 5 hours for PASTA, and as shown next, the

Table 2. Two iterations of PASTA compared to SATé on 50,000 RNASim given more than 24 hours of running time (outside TACC)

	Alignment Accuracy			Tree Error	Running Time
	SP-score	Modeler score	TC	FN	(hours)
PASTA-2iter	**80.2%**	**81.8%**	**311**	**8.2%**	10
SATé-2iter	20.5%	55.9%	30	12.6%	137

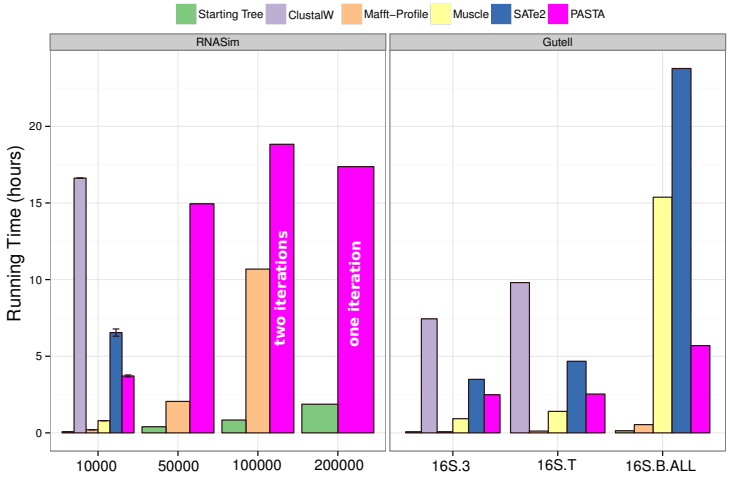

Fig. 4. Alignment running time (hours). Note that PASTA is run for three iterations everywhere, except on 100,000-sequence RNASim dataset where it is run for two iterations, and on the 200,000-sequence RNASim dataset where it is run for one iteration.

majority of SATé running time is spent in the merge step. However, the resulting SATé alignment is much less accurate, and produces trees that are substantially less accurate than PASTA (see Table 2).

Running Time. Figure 4 compares the running time (in hours) of different alignment methods. PASTA is faster than SATé, and MAFFT-Profile is faster than PASTA on the smallest datasets. However, the running time of MAFFT-Profile grows faster than PASTA so that at 200,000 sequences it is not able to finish in 24 hours, while PASTA can. Muscle is faster than PASTA on datasets with 10,000 sequences or less, but is slower on 16S.B.ALL, the only dataset above 10,000 sequence where it can actually run. Our approach for producing the starting tree is the fastest method on all datasets, and ClustalW is always the slowest. However, note that neither Clustalw or Muscle is parallelized and so these methods cannot take advantage of the multiple cores.

Figure 5a presents the running time comparison to SATé. Note that merging subset alignments is the majority of the time used by SATé to analyze the 50K RNASim dataset, but a very small fraction of the time used by PASTA.

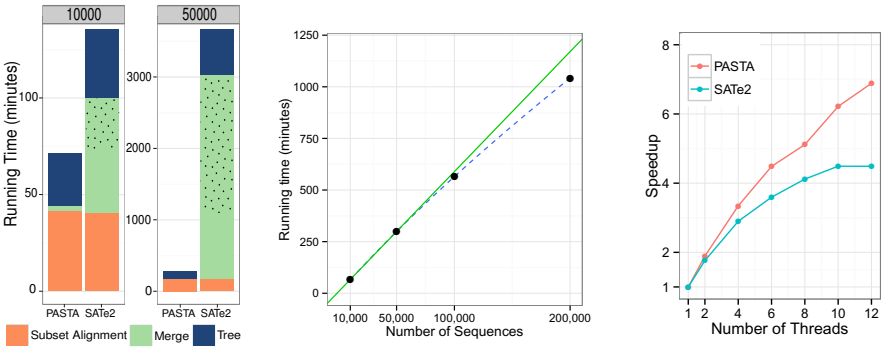

Fig. 5. Running time comparison of PASTA and SATé. (a) Running time profiling on one iteration for RNASim datasets with 10K and 50K sequences (the dotted region indicates the last pairwise merge). (b) Running time for one iteration of PASTA with 12 CPUs as a function of the number of sequences (the solid line is fitted to the first two points). (c) Scalability for PASTA and SATé with increased number of CPUs.

The reason SATé uses so much time is that all mergers are done hierarchically using either Opal (for small datasets) or Muscle (on larger datasets), and both are computationally expensive with increased number of sequences. For example, the last pairwise merge within SATé, shown by the dotted area in Figure 5a, is entirely serial and takes up a large chunk of the total time. PASTA solves this problem by using transitivity for all but the initial pairwise mergers, and therefore scales well with increased dataset size, as shown in Figure 5b (the sub-linear scaling is due to a better use of parallelism with increased number of sequences). Finally, Figure 5c shows that PASTA is highly parallelizable, and has a much better speed-up with increasing number of threads than SATé does. While PASTA has a much improved parallelization, it does not quite scale up linearly, because FastTree-2 does not scale up well with increased thread count.

Divide-and-Conquer strategy: impact of guide tree. We also investigated the impact of the use of the guide tree for computing the subset decomposition, and hence defining the Type 1 sub-alignments. We compared results obtained using three different decompositions: the decomposition computed by PASTA on the HMM-based starting tree, the decomposition computed by PASTA on the true (model) tree, and a random decomposition into subsets of size 200, all on the RNASim 10k dataset. PASTA alignments and trees had roughly the same accuracy when the guide tree was either the true tree or the HMM-based starting tree (Table 3). However, when based on a random decomposition, tree error increased dramatically from 10.5% to 52.3%, and alignment scores also dropped substantially. Thus, the guide-tree based dataset decomposition used by PASTA provides substantial improvements over random decompositions, and the default technique for getting the starting tree works quite well.

Table 3. Effect of subset decomposition in PASTA algorithm, based on one iteration of PASTA on one replicate of the 10k RNASim dataset

	Alignment Accuracy			Tree Accuracy
	SP-score	Modeler score	TC	FN
Random	78.4%	81.4%	2	52.3%
Phylogeny-based (estimated tree)	**86.3%**	**87.3%**	**138**	**10.5%**
Phylogeny-based (true tree)	85.5%	86.7%	133	**10.5%**

5 Discussion and Future Work

One of the intriguing observations in this study is that alignment accuracy measures are not always predictive of tree accuracy. For example, on the Gutell datasets, MAFFT-profile produced less accurate trees than Muscle, yet had better sum-of-pairs alignment accuracy scores. Similarly, the PASTA starting alignment is typically among the best in terms of alignment accuracy but far from the best in terms of tree error. Most likely this is because not all pairwise homologies are equally important for phylogeny estimation, and alignment accuracy measures treat pairwise homologies identically. Failing to recover some homologies may not have much impact on tree estimation, while other homologies may be essential for phylogenetic accuracy. Furthermore, alignment methods that aim to recover the conserved regions may be able to have high alignment accuracy scores but fail to produce good trees - because conserved regions may not be as useful for phylogeny estimation as regions that change. Thus, the sites and even specific homologies that are most informative of the phylogenetic branching process may not be the homologies that many alignment methods are trained to recover. More generally, then, this disconnect suggests a real challenge in using alignment metrics to predict the utility of an alignment, especially if the purpose of the alignment is phylogeny estimation.

We have shown results for the current default version of PASTA; however, we also explored variants where we changed some algorithmic parameters (see supplementary material [19]). We found PASTA to be robust to the choice of the starting tree. Interestingly, while varying the alignment subset size (between 50 and 200) had only a small impact on accuracy, PASTA run with smaller alignment subsets is much faster, raising the possibility that comparable accuracy at reduced running time might be achievable through smaller alignment subsets.

Finally, we note that PASTA, like SATé, is a method that "boosts" the performance (accuracy and/or scalability) of the base method used to align subsets. The good performance using MAFFT as the base method suggests the possibility that PASTA could be used to extend computationally intensive statistical methods, such as BAli-Phy [25], to large datasets, while maintaining their accuracy. Our future work will explore this possibility.

6 Conclusions

PASTA is a new method for nucleotide sequence alignment and tree estimation that is designed for speed, scalability, and accuracy, especially for large datasets.

PASTA is based on SATé, but its design allows it to provide improved accuracy while using only a fraction of the time on large datasets. The key algorithmic contribution is the new technique for aligning sequences on a given guide tree. This algorithmic design addresses computational limitations in SATé and other methods, however it also provides improved accuracy on large datasets because it uses transitivity to extend highly accurate overlapping alignments rather than trying to directly infer homologies between distantly related sets of sequences.

PASTA is fast and also scales well with the number of processors, so that datasets with even 200,000 sequences can be analyzed in less than a day with a small number of processors. Thus, highly accurate alignment and phylogeny estimation is possible, even on hundreds of thousands of sequences, without supercomputers.

PASTA software is implemented by extending the SATé code, and is publicly available at `https://github.com/smirarab/pasta`. Datasets are available at `http://www.cs.utexas.edu/users/phylo/software/pasta/`.

Acknowledgments. This research was supported in part by NSF grant DBI 0733029 to TW, by an International Predoctoral Fellowship to SM from the HHMI, and by a subgrant from the University of Alberta to TW, made possible through a donation from Musea Ventures, which is held by Professor Gane Ka-Shu Wong. The authors wish to thank the anonymous referees for their helpful comments. This research was supported in part by the iPlant Collaborative, NSF award number DBI-1265383.

A Appendix: Computing the Transitivity Merge

We compute the transitivity merge through a sequence of pairwise transitivity mergers. Recall that every node v in the spanning tree T^* computed in Step 2 is labelled by an alignment subset (i.e., a subset of the input sequence dataset on which we have a Type 1 sub-alignment). In addition, during Step 4, we computed Type 2 sub-alignments for every pair of Type 1 sub-alignments whose alignment subsets are adjacent in the spanning tree T^*. We now define a set $S(v)$ for every vertex v and $Label(e)$ for every edge e, as follows. For node v in T^*, we define the set $S(v) = \{X_v\}$ where X_v is the alignment subset associated to the node v, and for edge $e = (v, v')$, we set $Label(e) = (X_v, X_{v'})$. Note that $S(v)$ is a set containing one element - the alignment subset associated to v - and that $Label(e)$ is a pair of alignment subsets. Furthermore, we have computed Type 2 sub-alignments for each $X \cup Y$ where $Label(e) = (X, Y)$.

We will use T^* to guide a sequence of pairwise transitivity mergers, resulting finally in an MSA for the full set of sequences. As we do so, we will modify T^* through a sequence of edge contractions, until there is only one vertex left. The contraction of an edge $e = (v, w)$ will create a new vertex x with a new label

$S(x) = S(v) \cup S(w)$, but will not modify the labels at the edges. Therefore, at every point in the process, each edge will be labelled by a pair of alignment subsets for which we have a Type 2 sub-alignment, and each vertex will be labelled by a set of alignment subsets. Some edge contractions will require that we compute a transitivity merge of two overlapping compatible alignments. The new sub-alignments that result from transitivity mergers are called "Type 3 sub-alignments", and these Type 3 sub-alignments are defined by transitivity applied to some subset of the Type 2 sub-alignments.

For an edge $e = (v, w)$ and $Label(e) = (S_i, S_j)$, we have a Type 2 sub-alignment A^{ij} on $S_i \cup S_j$, and $S(v) \cap S(w) = \emptyset$. If $S(v)$ and $S(w)$ are singletons, then collapse the edge, and label the new vertex by the union of the labels at the endpoints. Otherwise, at least one endpoint of e is labelled by a set containing two or more alignment subsets, and the alignments A^v and A^{ij} are overlapping compatible alignments. Therefore, the three alignments A^{ij}, A^v, and A^w are all compatible, and so we can use transitivity (i.e., treating each alignment as an equivalence relation) to define the "transitivity merge" of these three alignments. To compute this transitivity merge, we first merge A^v and A^{ij}, and then we merge the resulting alignment with A^w (each step involves merging two overlapping compatible alignments). The result of each merger of these three MSAs creates a Type 3 sub-alignment on $S(v) \cup S(w)$. We contract the edge (v, w) to create the new node x, and we set $S(x) = S(v) \cup S(w)$.

Transitivity merge of two alignments. To compute the transitivity merge of two overlapping compatible alignments A and B, given two columns (one in A and the other in B) that share a common letter (i.e. the i^{th} character of the j^{th} sequence) we simply merge the two columns into one column.

Theorem 2. *Given m Type 1 alignments and $m-1$ Type 2 alignments, the algorithm to compute the transitivity merge of these alignments uses $O(Km \log m + mL)$ time, where K is the maximum length of any sequence (not counting gaps) in any Type 1 alignment and L is length of output alignment.*

Proof (Sketch): We begin with the following observation, which we provide without a proof due to space limitations (but see our supplementary material [19]). *Lemma: Let X, Y, and Z be disjoint sequence datasets, and alignments A and A' be alignments on $X \cup Z$ and $Y \cup Z$, respectively, that induce identical alignments on Z. Let K be the length of the longest sequence in X, Y, and Z, and L be the total number of sites in A and A'. Then we can merge alignments A and A' using transitivity in $O(L + (|X| + |Y| + |Z|)K)$.*

Let our dataset consist of N sequences, with each sequence of length at most K, and for the sake of simplicity, assume that our decomposition produces m subsets, all with equal sizes (note that centroid decomposition produces balanced subsets, so this assumption is justified). As described before, in Step 5, we chose an edge $e = (v, w)$ from the spanning tree, contract that edge, and perform two transitivity merges: one between $S(v)$ and $Label(e)$, and another between the result of the first merger and $S(w)$.

Based on the Lemma above, the first transitivity merge will have a running time of $O(K(|S(v)| + 2) + L)$, and the second merge will have a cost of $O(K(|S(v)| + |S(w)|) + L)$, and thus the cost of each edge contraction is $O(K(2 * |S(v)| + |S(w)|) + L)$. Now, imagine the case where the spanning tree is a path. If we start merging from one end to the other end, we get the total running time of $O(K(3 + 4 + \ldots + m)) = O(Km^2)$; however, we can improve on that. The important observation is that the spanning tree should be traversed such that transitivity mergers are between alignments with balanced number of sequences on each side.

The order in which edges are processed in PASTA is obtained by a recursive approach. Given the spanning tree, we divide it into two halves on the centroid edge, and thus obtain two roughly equal size subtrees. We process each half recursively using the same strategy, and thus get two single leaves at the endpoints of the centroid edge. Each leaf would represent the merger of all alignments in each half, and by construction they would have roughly equal size. We then contract the centroid edge, merge the two sides, and obtain the full alignment. If each half has roughly x sequences, the cost of the final edge contraction is $O(K(2x + x) + L) = O(3Kx + L)$ (as shown before). If $f(x)$ denotes the cost of applying our transitivity merger on a spanning tree with x nodes, we have

$$f(2x) = 2f(x) + 3kx + L$$

which has a $O(x \log(x) + xL)$ solution. Therefore, our particular order of traversing the spanning tree results in a total cost of $O(Km \log(m) + mL)$.

References

1. Sievers, F., Dineen, D., Wilm, A., Higgins, D.G.: Making automated multiple alignments of very large numbers of protein sequences. Bioinformatics 29(8), 989–995 (2013)
2. Liu, K., Raghavan, S., Nelesen, S., Linder, C.R., Warnow, T.: Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees. Science 324(5934), 1561–1564 (2009)
3. Liu, K., Warnow, T., Holder, M., Nelesen, S., Yu, J., Stamatakis, A., Linder, C.: SATé-II: Very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees. Syst. Biol. 61(1), 90–106 (2011)
4. Nelesen, S., Liu, K., Wang, L.S., Linder, C., Warnow, T.: DACTAL: divide-and-conquer trees (almost) without alignments. Bioinformatics 28(12), i274–i282 (2012)
5. Liu, K., Linder, C., Warnow, T.: Multiple sequence alignment: a major challenge to large-scale phylogenetics. PLoS Currents: Tree of Life (2010)
6. iPlant Collaborative: iPTOL, Assembling the Tree of Life for the Plant Sciences (2013), `https://pods.iplantcollaborative.org/wiki/display/iptol/Home`
7. Wong, G.K.S.: The Thousand Transcriptome (1KP) Project (2013), `http://www.onekp.com/project.html`
8. Katoh, K., Kuma, K., Toh, H., Miyata, T.: MAFFT version 5: improvement in accuracy of multiple sequence alignment. Nucl. Acids. Res. 33(2), 511–518 (2005)

9. Wheeler, T., Kececioglu, J.: Multiple alignment by aligning alignments. In: Proceedings of the 15th ISCB Conference on Intelligent Systems for Molecular Biology, pp. 559–568 (2007)
10. Edgar, R.C.: MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5(113), 113 (2004)
11. Edgar, R.C.: MUSCLE: a multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 32(5), 1792–1797 (2004)
12. Guo, S., Wang, L.S., Kim, J.: Large-scale simulating of RNA macroevolution by an energy-dependent fitness model. arXiv:0912.2326 (2009)
13. Price, M., Dehal, P., Arkin, A.: FastTree-2 approximately maximum-likelihood trees for large alignments. PLoS One 5(3), e9490 (2010)
14. Matsen, F., Kodner, R., Armbrust, E.: pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree. BMC Bioinformatics 11, 538 (2010)
15. Mirarab, S., Nguyen, N., Warnow, T.: SEPP: SATé-enabled phylogenetic placement. In: Pacific Symposium on Biocomputing, pp. 247–258 (2012)
16. Eddy, S.: A new generation of homology search tools based on probabilistic inference. Genome Inform. 23, 205–211 (2009)
17. Finn, R., Clements, J., Eddy, S.: HMMER web server: interactive sequence similarity searching. Nucleic Acids Research 39, W29–W37 (2011)
18. Mirarab, S., Warnow, T.: FastSP: Linear-time calculation of alignment accuracy. Bioinformatics 27(23), 3250–3258 (2011)
19. Mirarab, S., Nguyen, N., Warnow, T.: Supplementary Online Material, PASTA: ultra-large multiple sequence alignment. figshare (2014), `http://dx.doi.org/10.6084/m9.figshare.899770` (retrieved January 13, 2014)
20. Cannone, J., Subramanian, S., Schnare, M., Collett, J., D'Souza, L., Du, Y., Feng, B., Lin, N., Madabusi, L., Muller, K., Pande, N., Shang, Z., Yu, N., Gutell, R.: The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron and Other RNAs. BioMed. Central Bioinformatics 3(15) (2002)
21. Stamatakis, A.: RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinf. 22, 2688–2690 (2006)
22. Katoh, K., Frith, M.C.: Adding unaligned sequences into an existing alignment using MAFFT and LAST. Bioinformatics 28(23), 3144–3146 (2012)
23. Larkin, M.A., Blackshields, G., Brown, N.P., Chenna, R., McGettigan, P.A., McWilliam, H., Valentin, F., Wallace, I.M., Wilm, A., Lopez, R., Thompson, J.D., Gibson, T.J., Higgins, D.G.: Clustal W and Clustal X version 2.0. Bioinformatics 23(21), 2947–2948 (2007)
24. Boisseau, J., Stanzione, D.: TACC: Texas Advanced Computing Center (2013), `http://www.tacc.utexas.edu`
25. Suchard, M.A., Redelings, B.D.: BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22, 2047–2048 (2006)