Large-Scale Multiple Sequence Alignment and the Maximum Weight Trace Alignment Merging Problem

Paul Zaharias, Vladimir Smirnov, and Tandy Warnow

Abstract—MAGUS is a recent multiple sequence alignment method that provides excellent accuracy on large challenging datasets. MAGUS uses divide-and-conquer: it divides the sequences into disjoint sets, computes alignments on the disjoint sets, and then merges the alignments using a technique it calls the Graph Clustering Method (GCM). To understand why MAGUS is so accurate, we show that GCM is a good heuristic for the NP-hard MWT-AM problem (Maximum Weight Trace, adapted to the Alignment Merging problem). Our study, using both biological and simulated data, establishes that MWT-AM scores correlate very well with alignment accuracy and presents improvements to GCM that are even better heuristics for MWT-AM. This study suggests a new direction for large-scale MSA estimation based on improved divide-and-conquer strategies, with the merging step based on optimizing MWT-AM. MAGUS and its enhanced versions are available at https://github.com/vlasmirnov/MAGUS.

Index Terms—multiple sequence alignment, maximum weight trace, Markov clustering

1 Introduction

Multiple sequence alignment is a basic step in many bioinformatics pipelines, including phylogenetic estimation, protein structure prediction, the detection of selection, etc. Yet accurate alignment estimation is very challenging under some circumstances (e.g., when datasets evolve with high rates of evolution) [1]. Furthermore, only a few methods have been able to run on datasets with 1000 or more sequences [2], [3], [4], and many of the methods that can run on large datasets degrade in accuracy under those conditions [1], [4]. Thus, accurate alignment estimation on large datasets, especially ones that have evolved under high rates of evolution (and so have low pairwise sequence identity) is very challenging.

Divide-and-conquer strategies have been designed to enable highly accurate alignment methods to scale to large datasets while maintaining high accuracy. These methods operate in three stages: first the sequence dataset is divided into disjoint subsets, then alignments are computed for each of the subsets, and finally the subset alignments are merged into an alignment on the entire dataset. Examples of these methods include SATCHMO-JS [5], SATé [1], [6], and PASTA [2]. Of these, PASTA provides the best accuracy on datasets with many thousands of sequences and high rates of evolution, and has been shown to scale to 1,000,000 sequences. MAGUS [7] is a recent MSA method that uses this strategy, and it provides a substantial accuracy advantage over PASTA, especially on datasets with high rates of evolution. Yet MAGUS is essentially identical to PASTA for the first two stages, and only differs significantly in how it performs the third stage (where the disjoint subset alignments are merged into an alignment on the full dataset). Specifically, PASTA merges the set of k disjoint alignments by first merging $k-1$ pairs of alignments (using either OPAL [8] or Muscle [9]), and then uses the overlap between the merged pairs to define the final merged alignment on the basis of transitivity. In contrast, MAGUS uses a method called the Graph Clustering Method (GCM) to merge the k disjoint alignments all at once, rather than pairwise. Thus, the advantage MAGUS has over PASTA is due to the use of GCM instead of PASTA's technique.

This study aims to understand why the use of GCM produces highly accurate merged alignments. To do this, we formulate an optimization problem for merging disjoint alignments, which we called the Maximum Weight Trace Alignment Merging (MWT-AM) problem. The Maximum Weight Trace problem itself is a classical problem in bioinformatics, introduced by Kececioglu nearly 30 years ago [10].

The input to the MWT problem is a set of sequences (e.g., DNA sequences) and weights on pairs of letters from the different sequences, and the objective is an MSA that has the maximum total possible weight (defined to be the sum of the weights of aligned letters in the output MSA). Kececioglu [10] showed that MWT is NP-hard and can be exactly solved in $O(2^B L^B B^2)$ time for B sequences of total length L using dynamic programming. Kececioglu [10] also proposed a more practical branch-and-bound refinement of this algorithm, but the reduction in time is still not sufficient to analyze more than relatively small datasets (e.g., at most 20 sequences). Subsequent studies provided some additional techniques, both for exact solutions [11], [12] or heuristic solutions [13], [14]. Regrettably, not even the heuristics are able to run on even moderate-sized datasets, and the methods of Moreno and Karp [14] and Koller and Raidl [13] are not publicly available.

In this study, we define the **Maximum Weight Trace for Alignment Merger Problem**, or MWT-AM. We then

• *P. Zaharias, V. Smirnov and T. Warnow are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 61801.*
E-mail: warnow@illinois.edu

Manuscript received XXX XX, 2021; revised XXX XX, 202X.

show how GCM can be seen as a heuristic for this problem, applied to a particular input that is constructed by MAGUS, and we explore the optimization criterion scores it obtains. We then modify the GCM strategy to try to improve the MWT-AM scores, and we evaluate the correlation between MWT-AM scores and alignment accuracy. We prove that MWT-AM is NP-hard, and explore variants of GCM to improve the MWT-AM criterion scores.

Our study shows that the GCM technique and its variants are better solutions to MWT-AM than MAFFT-merge [15] and T-Coffee [16], which are the only other methods currently available that can merge disjoint alignments. Our study also shows that MWT-AM scores correlate well with alignment accuracy, suggesting that MWT-AM is a desirable optimization problem for use within divide-and-conquer alignment estimation. Finally, we show that using GCM and its variants within a divide-and-conquer pipeline enables analyses of datasets with thousands of sequences and produces highly accurate alignments. GCM and MAGUS are freely available at https://github.com/vlasmirnov/MAGUS.

2 Methods

2.1 Maximum Weight Trace Alignment Merging

The input to the Maximum Weight Trace (MWT) problem is a set S of sequences and a weight on selected pairs of letters from the sequences; the objective is a multiple sequence alignment of the sequences that maximizes the total weight of the pairs of letters that appear together in a column. The output alignment is also referred to as a "trace", which has a graph-theoretic description in [10] but can be more simply described as follows. A **trace** is a partition of the letters of the input sequences into an ordered set of pairwise disjoint subsets $X_1, X_2, \dots, X_k$ so that (1) each set has at most one letter from each sequence and (2) if letters x and y from the same sequence s appear in X_i and X_j respectively with $i < j$, then x appears before y in s. By treating the subsets as defining columns, these two properties ensure that the provided ordering of the subsets defines a multiple sequence alignment of the sequences. The **weight** of the trace is the sum of the weights of those pairs of letters that appear in the same subset, and hence in the same column in the alignment defined by the trace.

We generalize the MWT problem to allow the input to be a collection of disjoint alignments instead of a collection of individual sequences, and we refer to this as the Maximum Weight Trace Alignment Merging problem, or MWT-AM. We begin with some basic definitions.

Definition 1. Given an alignment A on sequence set S and a proper subset $S' \subset S$, the restriction of A to the sequences in S' is the sub-alignment of A induced by S'. In this restriction, if a site is entirely gapped in the sub-alignment, then it is removed (i.e., masked). Thus, the **induced** alignment contains no all-gap sites. Letting A' denote the induced alignment on S', we will say that A induces A'. Given a collection $\mathcal{A}$ of alignments $A_1, A_2, \dots, A_k$, where A_i is an alignment of set S_i, we say that A is a **merger** of the alignments in $\mathcal{A}$ if and only if A is an MSA of the sequences in set $S = \cup_i S_i$ and A induces A_i for each $i = 1, 2, \dots, k$. We will also refer to the alignments in $\mathcal{A}$ as **constraint alignments**.

Definition 2. Given an alignment A that is a merger of alignments in $\mathcal{A}$ and given columns c and c' drawn from different alignments, we let $A_{c,c'}$ be the indicator function that returns 1 if c and c' are in the same column in A and otherwise returns 0. Then, given a non-negative weight function $w(c, c')$, where c and c' are columns in different alignments in $\mathcal{A}$, we define the **weight** of A to be $weight(A) = \sum_{c,c'}[w(c, c') \times A_{c,c}]$. In other words, the weight of the merged alignment A is the sum of the weights of column pairs, where the two columns come from different alignments in $\mathcal{A}$.

We now define the MWT-AM problem, using these definitions. The input to MWT-AM is a set $\mathcal{A}$ of alignments as well as a set of weights on selected pairs of columns from different alignments in $\mathcal{A}$. The output will be an alignment A that is a merger of the alignments in $\mathcal{A}$ (see Definition 1) that maximizes the total weight (see Definition 2).

The MWT-AM problem can also be described in terms of the MWT problem, as we now show. A **trace** is a partition of the columns of the input constraint alignments into **an ordered list of** clusters, so that (1) each cluster contains at most one column from each alignment and (2) the ordering of the clusters is "valid": if columns x and y from the same constraint alignment A appear in clusters C_i and C_j, respectively, with $i < j$, then x appears before y in A. Such a trace trivially gives us a multiple sequence alignment that induces each of our constraint alignments (i.e, the trace defines a merged alignment). Then, given a weighting $w(x, y)$ for every pair of columns x, y from different constraint alignments, we can define the **weight** of a trace T to be $\sum_{x,y} w(x, y)$, where the sum is taken over all pairs of columns x, y belonging to the same cluster in T. It is easy to see that this is exactly the same as the weight of the merged alignment defined by the trace according to Definition 2.

We now formally define the MWT-AM problem:

Input: Multiple sequence alignments $A_1, A_2, \dots, A_k$, with A_i an MSA on set S_i of sequences with $S_i \cap S_j = \emptyset$ for $i \neq j$, and a weight function $w(x, y) \geq 0$ for every pair of columns x, y with x and y from different MSAs.
Output: a trace T on $A_1, A_2, \dots, A_k$ of maximum weight (i.e., a merged alignment of the input alignments that maximizes the total weight).

It is easy to see that if each MSA A_i is a single sequence, then MWT-AM is identical to MWT. It is also easy to see that finding a trace with the largest MWT-AM score is identical to finding a merged alignment with the total maximum weight (Definition 2).

Theorem 1. MWT-AM is NP-hard but can be solved in $O(2^B L^B B^2)$ for B alignments of total length L.

The proof follows easily from the corresponding theorems for MWT in [12].

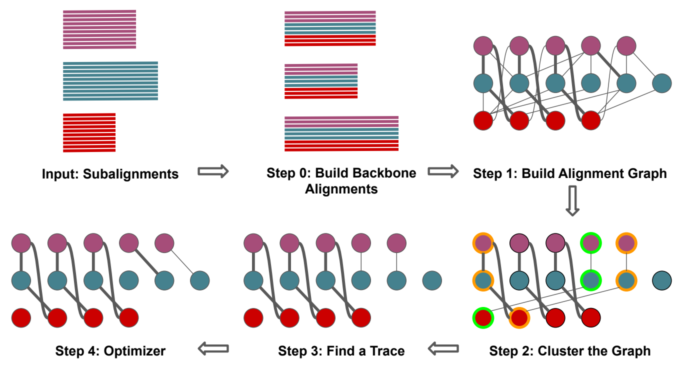

Input: Subalignments
Step 0: Build Backbone Alignments
Step 1: Build Alignment Graph
Step 2: Cluster the Graph
Step 3: Find a Trace
Step 4: Optimizer

Fig. 1: **The GCM algorithm design.** (Modified version of figure from [7].) The input to GCM is a set of constraint alignments. Optionally, we also supply weights on pairs of columns (sites) from different constraint alignments. **Step 0:** If the weights are not provided, this step constructs the backbone alignments, and uses these to compute the weights on pairs of columns. **Step 1:** We build an alignment graph, where each node represents a column from a constraint alignment and the weight of the edge between two nodes is the weight for the corresponding pair of columns. Thicker lines represent edges with higher weight. **Step 2:** We cluster the alignment graph with our method of choice. **Step 3:** We find a valid trace, where each cluster represents a column in our final alignment. Our example shows two common problems: the orange-outlined cluster contains columns from the same constraint alignment, and there is no valid ordering between the green-outlined cluster and the three other clusters that it "crosses". **Step 4:** We optionally run our trace through the optimizer, which will greedily move nodes between clusters to increase the MWT-AM score.

2.2 Graph Clustering Merger (GCM)

The Graph Clustering Merger (GCM) is an algorithm that was developed for use in MAGUS to merge the constraint alignments it produces in a divide-and-conquer setting. In this original formulation, GCM constructs a set of backbone alignments (i.e., a library graph) in order to weight the pairs of sites from different constraint alignments, but it can also be used to merge a set of disjoint alignments if the weights are provided by the user. In what follows, we describe GCM for use in both cases, noting that without user provided weights it will perform an initial step (referred to as Step 0 below) to compute the weights. The Graph Clustering Merger (GCM) then uses the weights it computes (or that it is given) to construct a merged alignment, which is referred to as a "trace" [10], using a sequence of steps. We explore variants of the original GCM algorithm from [7] in an attempt to improve the MWT-AM scores. Hence, the final general design for these GCM variants has several stages (Figure 1), some of which allow for variants. If the input includes weights on the pairs of sites from different constraint alignments, then GCM skips Step 0. Figure 1 presents a description of the stages of the algorithm.

2.2.1 *Variants on Step 2: Clustering*

The original technique for this step used in MAGUS is the Markov Clustering Algorithm (MCL) from [17]. Here we compare MCL to other approaches (for details, see Supplementary Materials, Section S1). Furthermore, within MCL, we vary the inflation factor to evaluate its impact.

Multi-level Regularized MCL (MRL-MCL) is a modification of MCL [18] and offers two extensions. The first is to improve MCL's scalability with a hierarchical structure: the input graph is coarsened to a more tractable size [19], clustered with MCL, uncoarsened, and the coarse clustering is refined. The second extension is a "regularization" operator, which is meant to smooth the flow distributions of neighboring nodes and reduce MCL's perceived over-clustering.

Region Growing (RG) is inspired by the heuristic in [13] and reminiscent of Kruskal's algorithm for finding minimum spanning trees. We initialize our clustering with every node in its own cluster. Each pair of clusters is added to a max heap, weighted by the weight of the edge between them. We then proceed to take pairs of clusters off the heap, merging the pair together if they don't contain any nodes from the same subalignment (i.e. the new cluster would be a valid MSA column). When we merge cluster B into cluster A, we update A's weight to all of B's neighbors (we call clusters "neighbors" if any of their elements have edges between them) as follows: for each cluster C among B's neighbors, we let $Weight(A, C) = Weight(A, C) + Weight(B, C)$ and put the pair (A,C) on the max heap. We continue merging pairs off the heap until we can't merge anything else.

2.2.2 *Variants for Step 3: Computing a trace*

We compare the original technique in GCM (which we will refer to as "minclusters") to two new techniques.

We explore the **Fiduccia-Mattheyses (FM)** algorithm [20], previously applied to the realm of multiple sequence alignment by MSARC [21]. FM is a heuristic for finding a minimum-weight split in a graph, constrained by a balancing requirement (the sizes of the two parts can differ by at most some value). Like MSARC, we use FM to recursively cut our graph in half, looking for bipartitions that minimize the weight of the split. Notably, the FM trace does not require a prior clustering, meaning that Step 2 can be skipped.

The **MWTgreedy** algorithm is basically a simple version of the first step of the algorithm described in the Moreno and Karp [14] paper. MWTgreedy takes the alignment graph (with or without clustering), and performs a breadth-first-search on the graph, starting from the "left"; when it finds a cycle, it breaks the lowest-weight edge in the cycle.

In **MWTSEARCH**, we combine mwtgreedy with the look ahead ability of the minclusters search algorithm used in MAGUS. Instead of just greedily breaking the smallest edge in every cycle, we consider every edge in the cycle as a possible move, and we try to find a path of such moves with the smallest weight.

The **RG-FAST** method is nearly the same as the Region Growing (RG) clustering method described earlier, but we constrain the algorithm to produce a valid trace, rather than just a clustering.

TABLE 1: Empirical Statistics of the Nucleotide Alignments. For the simulated datasets, the statistics are averaged across all replicates. We report the number of sites in the true (simulated) or reference (biological) alignment, the proportion of gaps, the alignment average gap length, the average normalized Hamming distance (ANHD), the maximum normalized Hamming distance (MNHD), and the number of replicate datasets.

Dataset	# seq	# sites	% gaps	Gap length	ANHD	MNHD	# reps.
RNASim	1000	4841.05	67.9	3.2	0.41	0.61	20
RNASim2	1000	4373.6	64.4	2.9	0.38	0.45	20
1000L1	1000	3817.5	73.2	13.6	0.70	0.77	20
1000L2	1000	2406.95	57.7	11.6	0.70	0.77	20
1000L3	1000	7042.75	85.2	20.0	0.69	0.76	20
1000L4	1000	2446.15	58.6	11.4	0.50	0.61	20
1000M1	1000	3965	74.4	10.1	0.70	0.77	20
1000M2	1000	3972.35	74.2	10.3	0.68	0.76	20
1000M3	1000	2722.55	62.8	7.6	0.66	0.74	20
1000M4	1000	2570.6	60.5	7.6	0.50	0.61	20
1000S1	1000	2141.15	53.0	4.0	0.69	0.77	20
1000S2	1000	1546	35.0	2.9	0.69	0.77	20
1000S3	1000	1595.25	37.0	2.9	0.69	0.76	20
1000S4	1000	1328.1	24.6	2.5	0.50	0.61	20
16S.M	740	2390	60.4	8.1	0.29	0.69	1
16S.3	5489	6646	77.4	7.4	0.29	0.70	1
16S.T	5548	8804	83	9.4	0.29	0.84	1
16S.B.ALL	24246	5328	72.9	3.5	0.21	0.46	1

2.2.3 *Variants for Step 4: Optimizer.*

This is an optional step, which uses a greedy strategy to improve the MWT-AM score, moving nodes between columns in our trace, and stopping when no additional gains can be made. A "move" entails transferring a node X from one cluster (column) to another, in such a way that the trace remains valid. In each iteration, we identify all valid moves (as defined above) with a positive improvement to the MWT-AM score. We perform these moves in descending order of gain, updating the gains of subsequent moves as needed. We stop when we no longer see any gainful moves.

2.3 Experimental Study

2.3.1 *Overview*

We explored GCM variants, T-Coffee [16], and MAFFT-merge [15]; to the best of our knowledge, T-Coffee and MAFFT-merge are the only other methods that are designed to merge three or more disjoint alignments. We used simulated and biological datasets from prior studies to explore these methods. The simulated datasets are nucleotide datasets that evolve down phylogenetic trees and so have true alignments, and the biological datasets are either nucleotide datasets from the Comparative Ribosomal Website [22] or protein datasets from the Homfam collection [23], which have curated reference alignments based on structure. We evaluated methods with respect to alignment accuracy and MWT-AM scores. For alignment accuracy, we used the number of shared homologies (i.e., the total number of true pairwise homologies recovered in the estimated alignment), SP-score (i.e., recall, or the fraction of true pairwise homologies recovered in the estimated alignment) and Modeler Score (i.e., precision, or fraction of the pairwise homologies in the estimated alignment that appear in the true or reference alignment), computed using FASTSP [24]. We also noted wallclock running times.

The analyses of simulated datasets and some of the smaller biological datasets were run on a single node with 16 CPUs and 64 GB of memory on the Campus Cluster at UIUC, and hence were limited to 4 hours. Those biological datasets that did not complete within 4 hours on the campus cluster were then run on the tallis cluster (which has 256 GB of memory) and allowed more time to complete (50 hours for the larger CRW datasets and 168 hours for the larger Homfam datasets).

For a given sequence dataset, we produce a collection of disjoint alignments and weights on the pairs of sites, as follows. Given a sequence dataset, we have two ways of producing inputs to GCM and other methods for MWT-AM: (1) We randomly decompose the dataset into disjoint subsets of approximately the same size (varying from 50 to 500) and (2) We use the default PASTA decomposition (mincluster) to produce the decomposition (i.e., PASTA computes an initial alignment and tree, and then decomposes the dataset into subsets using the tree by deleting edges until the subsets are of size at most 200). We then compute alignments on each subset using an existing method (default: MAFFT -l-ins-i [15]), thus producing a set of disjoint alignments. To define the weights on pairs of sites, we proceed as follows. We compute a set of 10 "backbone alignments", each of which contains 200 sequences that are obtained by selecting sequences randomly from the constraint alignments. These sequences are then aligned using existing methods (default: MAFFT -l-ins-i). These backbone alignments are then used to define the input to GCM, so that the weight on a pair of sites from different alignments is the weighted number of pairs of letters (one from each site) that appear in one or more of the backbone alignments (i.e., if x and y are two letters that are aligned in q backbone alignments, then this particular pair (x, y) contributes q to the total weight). As shown in Figure 2 in [7], this is one of the ways of defining the weights on pairs of sites that provide good final alignment accuracy (e.g., in general, accuracy is improved by using several backbones rather than just one, and by using larger backbones rather than smaller backbones).

2.3.2 *Experiments*

We performed three experiments.

- Experiment 0 explores the correlation between MWT-AM scores and alignment accuracy.
- Experiment 1 explores variants of the GCM-MWT algorithm in order to optimize MWT-AM scores.
- Experiment 2 compares the best variants of GCM-MWT in comparison to T-Coffee and MAFFT-merge, the other methods that can merge disjoint alignments.

Experiments 0 and 1 use the training datasets (1000M1 and 1000M4), and Experiment 2 uses the remaining datasets (i.e., ”testing datasets”).

2.3.3 *Datasets*

We studied performance on both biological nucleotide and proteins) and simulated datasets, all from prior studies (and available online). In total, we analyzed 304 datasets (284 nucleotide datasets and 20 protein datasets), and performed multiple analyses on each.

Simulated data We use a total of 14 model conditions, each with 1000 sequences. 12 of these were used in the papers introducing SATé and PASTA [1], [2], [6]: we use 1000M1 and 1000M4 for the training phase, and then 1000L3, 1000S1, 1000M2, 1000L1, 1000S2, 100S3, 1000M3, 10000L2, 1000L4, and 1000S4 for the testing phase. These sequences were evolved using the ROSE [25] software, and evolve under *i.i.d.* evolution with substitutions and indels, and under three indel lengths –long (L), short (S), and medium (M). The integer at the end of the model condition roughly with respect to the rate of evolution, with 1 indicating the highest rate of evolution. Each model condition has 20 replicates. We also use two subsets of the RNASim [2] simulated datasets, named RNASim1 and RNASim2, each with 1000 sequences. The RNASim datasets evolve under a model that includes selection and takes RNA secondary structure into consideration. We provide the empirical statistics for these simulated nucleotide datasets in Table 1.

CRW (nucleotide) datasets We use four nucleotide datasets developed by Robin Gutell for the Comparative RNA Website [22], and which were used in previous studies to evaluate alignment accuracy [2], [3], [6], [7]. Specifically, we used versions of the 16S.M, 16S.3, 16S.T, and 16S.B.ALL datasets from [7], which were used to evaluate MAGUS, and obtained by taking the corresponding datasets from [6] and then pruning them to remove sequences that are more than 20% from the median sequence length. Three of the four CRW datasets have at least 5000 sequences and the remaining one has only 740 sequences. We provide the empirical statistics for these CRW datasets in Table 1.

HomFam (protein) datasets We also used 20 protein datasets from the Homfam [23] collection. These range in size from about 10,000 sequences to nearly 94,000 sequences, and each has a reference alignment based on protein structure for a small subset of its sequences.

2.3.4 *Evaluation criterion*

For alignment accuracy, we report number of shared homologies (i.e., the total number of true pairwise homologies recovered in the estimated alignment), SP-score (i.e., recall, or the proportion of pairwise homologies in the reference alignment that appear in the estimated alignment) and Modeler score (i.e., precision, or the proportion of pairwise homologies in the estimated alignment that appear in the reference alignment). We also report the MWT-AM score, using the backbone alignments to define the weights on pairs of columns from the constraint alignments.

3 Results

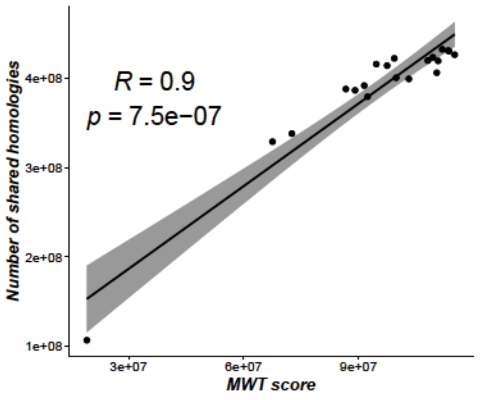
3.1 Results for Experiment 0: Correlation analyses

We explored the correlations between MWT-AM criterion and alignment accuracy using the default version of GCM on 1000M1 and 1000M4, and under two ways of decomposing the dataset: random decompositions and decompositions obtained during the first iteration of a default PASTA alignment.

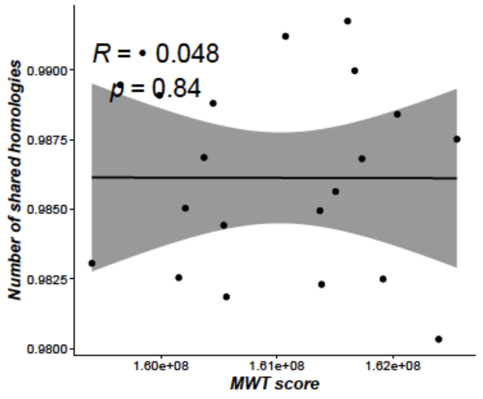
Correlations using GCM with random subset decompositions. We ran default GCM on the 1000M1 and 1000M4 datasets using random subset decomposition with subsets of size 50, 100, 200 and 500. The number of backbones was set to 1 and 10, with sizes of 20, 40, 100, 200 and 500. For each model condition, we reported MWT-AM scores and the number of shared homologies between the estimated and the true alignment, and then Spearman’s ρ was computed on all 20 replicates (Supplementary Table S1). These results show correlations ranging from 0.851 to 0.997 for 1000M1 and from 0.241 to 0.971 for 1000M4. Conditions for which 1000M4 returned low correlations correspond to ’extreme’ conditions with only 1 backbone and 20 sequences. Thus, the correlation is very strong for 1000M1, a model condition where alignment is challenging so that SP-scores are not very high, and lower (but moderately strong) for 1000M4, a model condition where alignment is easier and SP-scores are high. Furthermore, the correlations are statistically significant ($p < 0.005$) for all conditions except for one ’extreme’ condition.

Correlations using GCM with default PASTA decomposition. We ran default version of GCM from [7] within the standard divide-and-conquer pipeline described above (10 backbones each containing 200 sequences). Results (Figure 2) show that the correlations between MWT-AM scores and alignment accuracy (using the number of shared homologies) are very strong for the 1000M1 model condition (ρ=0.9) and null (ρ = -0.048) on the 1000M4 model condition. For 1000M1, the Spearman rank correlation is statistically significant (Figure 2) but not for 1000M4. Increasing the size or number of the backbone alignments also improves accuracy and strengthens the correlation, as shown in Supplementary Table S1 (for random decompositions) and Supplementary Figure S2 (for PASTA decompositions).

Thus, for most model conditions (1000M1 and 1000M4) using random decomposition and specifically for a hard condition (1000M1) using a default PASTA decomposition, the Spearman rank correlation coefficient between MWT-AM scores and alignment accuracy (measured using the number of shared homologies) is at least moderately high



(a) 1000M1



(b) 1000M4

Fig. 2: Correlation analysis of default GCM MWT-AM scores and number of shared homologies on (a) 1000M1 and (b) 1000M4. For each analysis, the subsets are obtained using a first iteration PASTA decomposition. We show Spearman's ρ. The x-axis corresponds to the MWT-AM score and the y-axis corresponds to the number of shared homologies of GCM's output alignment with respect to the true alignments.

and statistically significant. This suggests strongly that optimizing MWT-AM scores is a valuable approach to alignment estimation.

3.2 Results for Experiment 1: Designing GCM-MWT

Impact of varying MCL inflation factor within MCL. We explored variants of GCM when using MCL for Step 2 in which we varied the inflation factor (IF) parameter within MCL, which controls the granularity of the clustering. We ran GCM varying IF between 1.1, 1.4, 2, 3, 4, 6, 8 and 10, and recorded the MWT-AM scores (supp. Fig. S1). These experiments reveal that selecting any IF in the range 2–10 produce equivalently good results, and lower IF values produce worse MWT-AM scores. We select IF=4.0 for the default setting.

Exploring other aspects of GCM. We explore all the variants of GCM described on our training datasets, 1000M1 and 1000M4. Four different methods are available for step 2 (MCL, MLR-MCL, RG), six options for step 3 (minclusters, FM, mwtgreedy, mwtsearch, RG, RG-FAST) and step 4 has two possibilites (using or not using the optimizer). In addition, some steps are not mandatory for some combinations; step 4 is optional, and step 2 can be omitted if FM, mwtgreedy, mwtsearch, RG and RG-FAST are used in step 3. In all, there are currently 34 possible combinations of steps 2–4 for the complete GCM pipeline.

A comparison between these variants with respect to MWT-AM score on the 1000M1 condition (Table 2) reveals three main outcomes. First, very few combinations on the default combination implemented in GCM (step 2: MCL, step 3: minclusters, step 4: nothing) resulted in an improvement in MWT-AM scores, and any improvement that resulted was very small. Second, some combinations showed much less accurate results than default mode (e.g., RG+MWTGREEDY), but all combinations are more or less even (and very slightly better than the default combination) in terms of average accuracy when step 4 (the optimizer) is added. This is not surprising, since the optimizer is designed to increase the MWT-AM score. Results on the 1000M4 condition show very little difference between variants, indicating that this model condition is generally too easy to distinguish between variants (Supp. Table S2).

We note that improvements in the MWT-AM score tend to also result in an improvement in alignment accuracy using SP-score, a trend that is consistent with our earlier observations. Interestingly, improvements in SP-score resulting from using the optimizer tend to result in decreases in the Modeler score, but the average of these two alignment accuracy measures either stays the same or improves.

Among the variants that do not use the optimizer step, we picked the default setting from MAGUS (i.e., MCL+minclusters) as one of the GCM variants to explore further in our testing phase: it ties for best for both MWT-AM score and alignment accuracy (average of SP-score and Modeler score) of all the methods that complete on all the datasets. There is essentially no difference between methods that use the optimizer score (except for running time, where some methods are more computationally intensive); therefore, we picked FM+OPTIMIZER as a second variant to pick, in the hope that the substantial difference in approach might reveal some benefits during the testing phase. In sum, we selected two variants for GCM to explore in the testing phase:

- `MCL+minclusters` (i.e., no optimizer step at the end); note that this is the default setting used in MAGUS [7], and so we also refer to this as GCM(default).
- `FM+OPTIMIZER`; we refer to this as GCM(fm-opt).

3.2.1 Results for Experiment 2: Comparing GCM-MWT to other methods

We compare the GCM variants to MAFFT–merge (two variants) and T-Coffee with respect to alignment accuracy (SP-score, Modeler score, and the average of these two) and running time when applied to a set of disjoint alignments. For GCM, the weights are computed from the backbone alignments it computes. In these experiments, T-Coffee is

TABLE 2: Results of Experiment 1 on 1000M1. Median scores and running times where computed on 19 replicates. Stars (*) indicate a step for which one or more replicates didn't finish within 4 hours of running time on campus cluster (nodes with 16 cores, 64GB of memory); in these cases, we report median across the replicates that completed.

Algorithms			Scores (Accuracy %)				Running time (sec)			
Step 2	Step 3	Opt.	MWT	SP	Modeler	Avg (SP,Mod)	Step 2	Step 3	Opt.	Total
mcl	minclusters		99,812,063	83.6	86.5	85	8	0		27
mcl	fm		91,014,447	74.9	85.5	80.2	6	5		21
mcl	mwtgreedy		99,052,955	83	86.7	84.9	6	0		17
mcl	mwtsearch		99,052,955	83	86.7	84.9	6	0		17
mlrmcl	fm		90,471,403	74.6	85.4	80	7	9		33
mlrmcl	minclusters		99,454,384	83.3	86.7	85	8	0		22
mlrmcl	mwtgreedy		99,215,451	82.7	86.7	84.7	8	0		25
mlrmcl	mwtsearch		99,215,793	82.7	86.7	84.7	8	0		24
	fm		94,057,633	76.3	85.9	81.1	0	26		36
	mwtgreedy		100,135,039	83.7	86.6	85.2	0	945		957
	mwtsearch*		101,576,296	83.8	86.9	85.3	0	2198		2210
	rg		92,466,287	79.7	85.9	82.8	0	13		24
	rgfast		84,047,187	70.3	86.3	78.3	0	5		16
rg	fm		91,372,117	76	85.7	80.8	18	11		45
rg	minclusters		90,993,911	75.9	86.7	81.3	14	131		159
rg	mwtgreedy		67,580,556	59.3	87.7	73.5	18	1		37
rg	mwtsearch		68,047,192	58.7	87.7	73.2	15	95		115
mcl	minclusters	opt	100,245,267	83.7	86.4	85	6	0	17	33
mcl	fm	opt	100,254,633	83.6	86.4	85	6	6	31	54
mcl	mwtgreedy	opt	100,246,504	83.7	86.4	85.1	6	0	24	39
mcl	mwtsearch	opt	100,246,504	83.7	86.4	85.1	6	0	28	46
mlrmcl	fm	opt	100,208,349	83.6	86.3	85	5	5	26	46
mlrmcl	minclusters	opt	100,210,334	83.6	86.3	85	5	0	14	30
mlrmcl	mwtgreedy	opt	100,210,967	83.6	86.4	85	5	0	21	37
mlrmcl	mwtsearch	opt	100,210,967	83.6	86.4	85	5	0	19	36
	fm	opt	100,179,905	83.8	86.5	85.1	0	25	35	71
	mwtgreedy	opt	100,207,425	83.8	86.5	85.1	0	947	25	984
	mwtsearch	opt*	103,371,769	84	86.8	85.4	0	2112	22	2135
	rg	opt*	101,807,628	82.9	85.6	84.2	0	12	20	46
	rgfast	opt	100,201,843	83.7	86.4	85	0	6	48	65
rg	fm	opt	100,199,665	83.6	86.3	85	11	8	28	59
rg	minclusters	opt	100,159,588	83.7	86.4	85	11	121	29	182
rg	mwtgreedy	opt	100,221,006	83.6	86.3	85	11	1	114	144
rg	mwtsearch	opt	100,219,555	83.6	86.3	85	11	107	122	203

much less accurate than the other methods (supp. Figure S3). Based on a discussion with the developer, Cedric Notredame, T-Coffee is not designed for datasets of this size and has also not been tested sufficiently on nucleotide datasets. Hence the use of T-Coffee for merging large nucleotide alignments may not be appropriate.

The rest of this section focuses on the comparison between the GCM variants and MAFFT–merge, using default decompositions (which are phylogeny-based) or random decompositions.

Results on random subset decompositions. We selected three simulated datasets—1000L3, RNASim and 1000S4—and tested the performance of GCM(default) and MAFFT–merge (using the most accurate version based on L-ins-i) when the constraint alignments are obtained through random decomposition rather than the phylogeny-based mincluster decomposition.

We compared the default GCM (MCL+minclusters) and MAFFT-merge on random decompositions, using subsets of size 50, 100, 200 and 500. The number of backbones was set to 1 and 10, with sizes of 20, 40, 100, 200 and 500. For each model condition, we report SP-scores.

Results for alignment accuracy (Figure 4) show that MAFFT–merge and GCM(default) have close accuracy for slowly evolving sequences (1000S4 and RNASim) but GCM(default) is more accurate on fast-evolving sequences (1000L3). This is similar to what we observed on the PASTA decompositions in Figure 3. Surprisingly, MAFFT–merge performed relatively well despite not having the advantage of using alignments on local subsets as constraints (see recommendations for using MAFFT–merge at https://mafft.cbrc.jp/alignment/software/merge.html).

However, a comparison between these results and the same methods with the default phylogeny-based decomposition in Figure 3 shows that random decompositions produce worse alignments. This underlies the importance of the subset decomposition step in pipelines that use divide-and-conquer to estimate alignments.

Results using default decompositions on simulated datasets. In Figure 3, we compare the two variants of MAFFT–merge to GCM(default) and GCM(FM+opt) on all the testing model conditions with default decompositions. This comparison shows the following trends. The two ways of running MAFFT–merge are clearly distinguished, with the L-ins-i version much more accurate than the default version. Both ways of running GCM are approximately equal in accuracy, and both are almost always more accurate than MAFFT–merge using L-ins-i, with the only exceptions being the easiest datasets (1000L4 and 1000S4) that have the lowest rate of evolution.

Running time on simulated datasets using default decompositions. MAFFT–merge in default mode is the

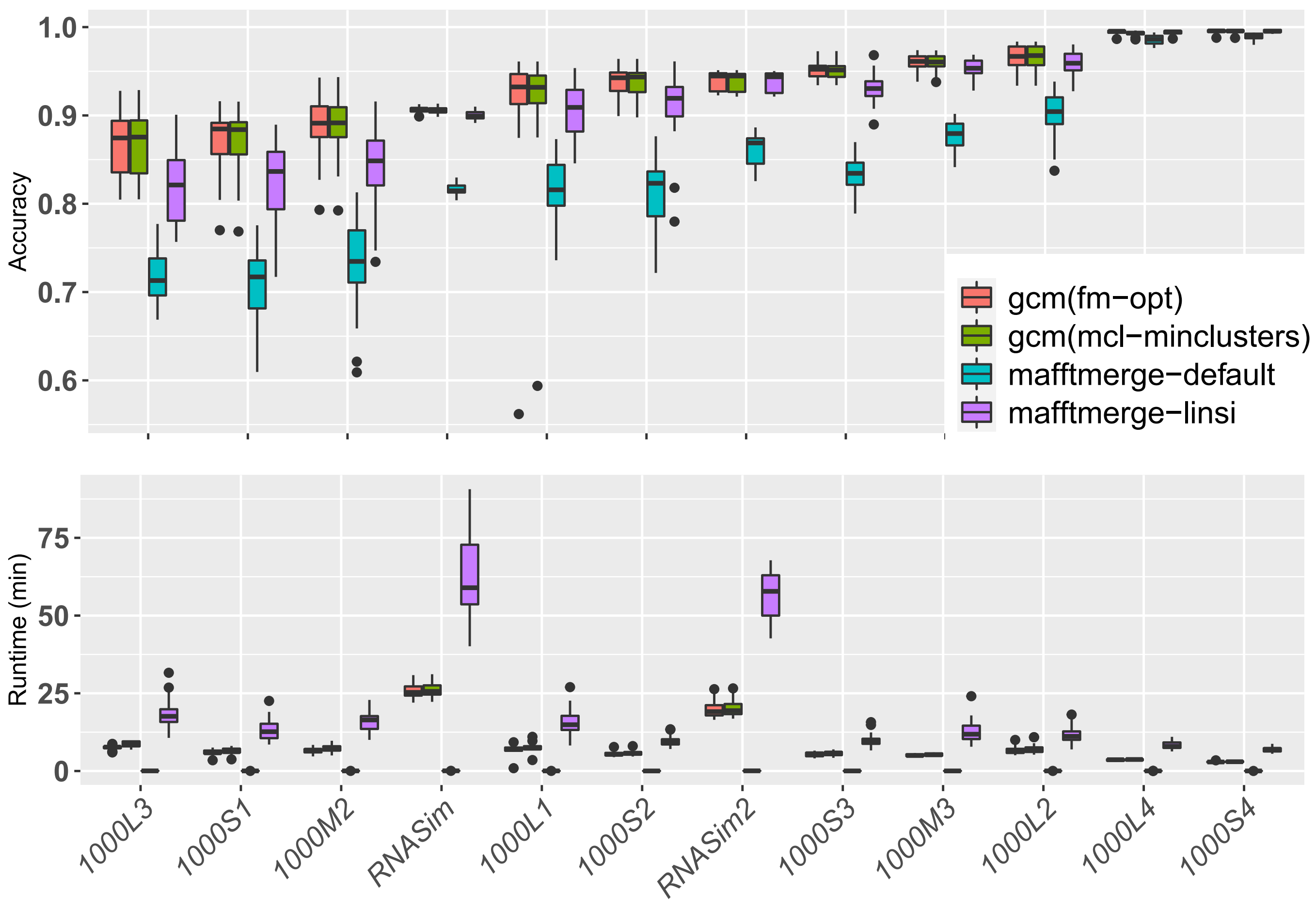


Fig. 3: Experiment 2: Results on the simulated datasets using default decompositions (phylogeny-based), without T-Coffee. Accuracy refers to the average of SP-score (i.e., recall) and Modeler score (i.e., precision) in the estimated alignments. Running time is based on wallclock time, using the Campus Cluster at the University of Illinois constrained to machines with the same amount of memory. All analyses here were performed on datasets produced in the first iteration of a default PASTA analysis.

fastest method, followed by both GCM(default) and GCM(FM+opt), and then by MAFFT–merge using L-ins-i (Figure 3). Both ways of running GCM are often twice as fast as running MAFFT–merge using L-ins-i, and finish on all these datasets in at most 30 minutes.

Results on biological nucleotide (CRW) datasets. Table 3 shows results on the CRW datasets. Although the methods are very similar on the smallest dataset, the methods are clearly distinguishable on the larger datasets. Most importantly, MAFFT–merge is less accurate than both GCM variants, obtaining SP-scores that are 10–13% lower and Modeler scores that are 4–6% lower than GCM(default). It is important to realize that the L-ins-i option for MAFFT–merge is unable to complete in 4 hours on these larger datasets, and so we used the default setting for MAFFT–merge, which is less accurate. A comparison between the two GCM variants shows that they have the same average alignment accuracy (where we average SP-score and Modeler score), but GCM(default) has somewhat better Modeler score and GCM(fm+opt) has somewhat better SP-score.

A comparison of running times (Table 3) shows the following trends. On the 16S.M dataset (with 740 sequences) we used MAFFT–merge with the L-ins-i setting, which is more accurate but also more expensive; as a result, the three methods have nearly identical running times (6 minutes). The other three datasets are larger, requiring that we use the default version of MAFFT-merge instead of the L-ins-i version. On these three larger datasets, MAFFT–merge is much faster, finishing in under a minute on 16S.3 and 16S.T (each with about 5500 sequences), and in 7 minutes on the 16S.B.ALL dataset (with 24,246 sequences); in contrast, the two GCM variants take longer. As expected, GCM(default) is faster, with 12–15 minutes on all three datasets, but GCM(fm+opt) is slower: 31–38 minutes on 16S.3 and 16S.T, and then unable to complete on the 16S.B.ALL dataset within the allowed time (48 hours).

Results on biological protein (Homfam) datasets.

We also compared GCM-default, GCM(fm+opt), and MAFFT–merge on several large Homfam datasets with up to nearly 94,000 sequences.. MAGUS and recursive MAGUS were evaluated on these datasets in comparison to many established protein alignment methods (e.g., MAFFT, Clustal-Omega, and Muscle) as well as more recently developed alignment methods (UPP and PASTA) in [26], and shown

TABLE 3: Experiment 2: Results on the CRW datasets; the best results are boldfaced (ties are for methods within 1% of the best found result). N.A. means "not applicable"

Dataset (# seqs)	Method	MWT score	SP-Score	Modeler	Avg.	Time (mins.)
16S.M (740)	gcm (default)	**137,370,948**	**88.4**	**86.9**	**87.7**	5.8
	gcm (fm+opt)	**137,487,088**	**88.4**	**86.7**	**87.6**	5.9
	mafftmerge (l-ins-i)	N.A.	**88.5**	**87.0**	**87.7**	6.0
	mafftmerge (default)	N.A.	85.9	86.7	86.3	0.02
16S.3 (5,489)	gcm (default)	247,066,576	**92.1**	**86.5**	**89.3**	12.4
	gcm (fm+opt)	**249,384,030**	**92.8**	**85.9**	**89.3**	31.6
	mafftmerge (l-ins-i)	N.A.	91.6	**85.8**	88.8	2926.5
	mafftmerge (default)	N.A.	81.7	81.4	81.5	0.5
16S.T (5,548)	gcm (default)	244,923,269	**92.4**	**87.7**	**90.1**	15.0
	gcm (fm+opt)	**246,839,091**	**92.7**	86.4	**89.5**	37.7
	mafftmerge (l-ins-i)	N.A.	91.4	86.2	88.8	2607.1
	mafftmerge (default)	N.A.	81.7	82.3	82.0	0.6
16S.B.ALL (24,246)	gcm (default)	139,531,511	**95.8**	**95.7**	**95.7**	13.8
	mafftmerge (default)	N.A.	83.4	90.5	87.0	7.0

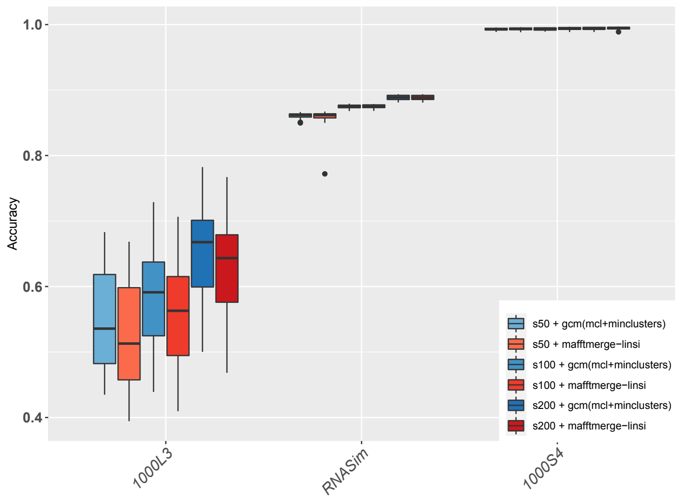


Fig. 4: Experiment 2: Comparison between MAFFT-merge and GCM(default) on random decompositions on three simulated model conditions. Random subsets of size 50, 100 and 200 were constructed. Accuracy refers to the average of SP-score (i.e., recall) and Modeler score (i.e., precision) in the estimated alignments.

to be more accurate (meaning, having lower average of SPFN and SPFP) than all of them, with MAGUS more accurate than recursive MAGUS. Given the dominance of MAGUS in that study, here we focus the comparison on variants of MAGUS where we modify the merger step (i.e., replacing GCM-default, as used in MAGUS, with GCM+opt and MAFFT–merge(default)). We allowed all methods to run up for a a full week (168 hours) on the tallis cluster, which has 256 GB of memory.

These results show the following clear trends. First, in every dataset, GCM(fm+opt) produces better MWT-AM scores and also better SP-scores than GCM-default (Table 4), showing that optimizing MWT-AM scores correlates with improved SP-scores on these datasets as well. Second, both GCM-default and GCM(fm+opt) complete on every dataset, whereas mafftmerge(default) fails to return alignments on the two largest datasets (zf-CCHH with 88,345 sequences and rvp with 93,681 sequences). For those two cases, MAFFT returns a warning that some groups are forced to be a monophyletic cluster and then uses a huge computational time reallocating sequences into sub-MSAs. Third, when restricted to the remaining 18 datasets where MAFFT-merge(default) could run, we see that GCM(fm+opt) came had the best SP-scores in 11 datasets and MAFFT-merge(default) had the best SP-scores in 7 datasets. The average across the 18 datasets also show an advantage for SP-score to GCM(fm+opt).

4 Discussion

This study showed that the MWT-AM optimization criterion (as computed in this study, using backbone alignments to define the weights) is generally highly correlated with alignment accuracy measured using the number of shared homologies. This trend shows that optimizing the criterion is desirable. This study also showed that the default usage of GCM within MAGUS is an effective heuristic for MWT-AM, explaining why MAGUS produces highly accurate merged alignments. In other words, by posing and studying MWT-AM, we are able to explain why MAGUS is a highly accurate alignment method.

On the other hand, we also saw that some modifications to the GCM heuristic, largely through the use of a final "optimizer" stage, can improve the MWT-AM scores compared to GCM-default (the use of the technique as performed in MAGUS). Furthermore, both GCM(default) and GCM(FM+opt) are sufficiently fast that they can be used to merge a large number of alignments; hence, both ways of using GCM are effective within divide-and-conquer pipelines for large-scale multiple sequence alignment.

A comparison to other methods that can merge disjoint alignments reveals some significant differences. As we have noted, T-Coffee did not produce satisfactory results in these experiments, but the explanation is that T-Coffee has been designed primarily for amino acid alignments rather than nucleotide alignments, and has not been tested in this setting. Given the outstanding accuracy of T-Coffee that has been observed for amino acid alignments (e.g., [27]), there is an opportunity for future work to provide a version of T-Coffee explicitly for merging large nucleotide alignments.

The comparison to MAFFT–merge is also informative, and shows that MAFFT–merge using L-ins-i per-

TABLE 4: Experiment 2: Results on the Homfam datasets; the best results are boldfaced (ties are for methods within 1% of the best found result). Average shown without the rvp and zf-CCHH datasets, because mafftmerge did not finish on these datasets (indicated by NaN). N.A. means "not applicable".

Dataset (# seqs)	Method	MWT score	SP-Score	Modeler	Avg.	Time (min.)
aat (25,100)	gcm (default)	29,664,610	78.1	**89.5**	**83.8**	13.2
	gcm (fm+opt)	**30,096,177**	**82.8**	86.6	**84.7**	674.5
	mafftmerge (default)	N.A.	61.6	73.9	67.8	**2.1**
Acetyltransf (46,285)	gcm (default)	20,487,572	51.8	**87.3**	**69.5**	554.5
	gcm (fm+opt)	**24,717,427**	**56.6**	65.4	61.0	608.0
	mafftmerge (default)	N.A.	51.1	62.0	56.5	**6.4**
adh (21,331)	gcm (default)	7,933,894	58.6	96.5	77.5	3.4
	gcm (fm+opt)	**8,029,439**	60.9	**98.6**	79.7	113.7
	mafftmerge (default)	N.A.	**93.7**	96.1	**94.9**	**1.3**
aldosered (13,277)	gcm (default)	26,234,848	88.8	**95.6**	**92.2**	4.5
	gcm (fm+opt)	**26,783,951**	**91.2**	94.3	**92.7**	101.0
	mafftmerge (default)	N.A.	82.9	88.3	85.6	**0.7**
biotin_lipoyl (11,833)	gcm (default)	6,875,466	90.4	**95.9**	93.1	0.1
	gcm (fm+opt)	**6,942,195**	92.5	93.1	92.8	4.1
	mafftmerge (default)	N.A.	**95.5**	94.2	**94.9**	**0.3**
blmb (17,200)	gcm (default)	5,181,458	55.1	**81.2**	68.1	12.9
	gcm (fm+opt)	**5,774,807**	**68.2**	74.0	**71.1**	184.9
	mafftmerge (default)	N.A.	55.4	69.7	62.5	**0.8**
ghf13 (12,607)	gcm (default)	13,803,396	58.4	**88.7**	**73.6**	13.8
	gcm (fm+opt)	**14,823,697**	**61.2**	80.6	70.9	123.8
	mafftmerge (default)	N.A.	49.8	70.2	60.0	**0.6**
gluts (10,099)	gcm (default)	10,717,250	60.1	**91.4**	**75.8**	2.6
	gcm (fm+opt)	**11,195,201**	61.5	83.0	72.2	20.4
	mafftmerge (default)	N.A.	**70.8**	79.8	**75.3**	**0.2**
hla (13,465)	gcm (default)	17,099,545	**99.8**	**100**	**99.9**	**0.3**
	gcm (fm+opt)	**17,106,544**	**100**	**100**	**100**	26.8
	mafftmerge (default)	N.A.	**99.3**	**100**	**99.7**	0.5
hom (12,037)	gcm (default)	5,080,252	94.1	**97.9**	**96.0**	**0.1**
	gcm (fm+opt)	**5,092,477**	**94.5**	96.8	**95.6**	5.3
	mafftmerge (default)	N.A.	**95.3**	96.0	**95.6**	0.2
myb_DNA-binding (10,398)	gcm (default)	7,080,357	89.0	**96.5**	**92.7**	**0.2**
	gcm (fm+opt)	**7,273,310**	90.2	93.2	91.7	3.8
	mafftmerge (default)	N.A.	**92.8**	92.8	**92.8**	**0.2**
p450 (21,013)	gcm (default)	14,229,615	41.7	**85.4**	63.6	762.5
	gcm (fm+opt)	**16,186,172**	**58.9**	78.7	**68.8**	937.4
	mafftmerge (default)	N.A.	53.5	70.6	62.1	**2.2**
PDZ (14,950)	gcm (default)	9,354,085	80.2	**90.0**	**85.1**	1.2
	gcm (fm+opt)	**9,732,891**	**83.4**	88.8	**86.1**	30.3
	mafftmerge (default)	N.A.	74.1	81.3	77.7	**0.5**
Rhodanese (14,049)	gcm (default)	10,128,517	66.2	**88.5**	**77.3**	1.7
	gcm (fm+opt)	**10,683,589**	**71.9**	79.1	75.5	33.4
	mafftmerge (default)	N.A.	48.0	60.1	54.1	**0.5**
rhv (17,976)	gcm (default)	4,142,652	25.3	**89.5**	**57.4**	4.2
	gcm (fm+opt)	**4,315,175**	26.2	43.0	34.6	76.5
	mafftmerge (default)	N.A.	**32.1**	57.2	44.6	**0.9**
rrm (27,610)	gcm (default)	7,713,839	76.0	**86.7**	**81.4**	**1.5**
	gcm (fm+opt)	**7,751,988**	**78.3**	81.4	79.9	45.6
	mafftmerge (default)	N.A.	72.0	78.8	75.4	1.8
rvp (93,681)	gcm (default)	146,105,340	82.5	**86.0**	84.2	3982.0
	gcm (fm+opt)	**146,115,355**	**85.7**	**86.2**	**86.0**	**862.8**
	mafftmerge (default)	N.A.	NaN	NaN	NaN	NaN
sdr (50,157)	gcm (default)	63,658,307	62.4	**90.5**	**76.5**	26.1
	gcm (fm+opt)	**65,182,563**	63.3	88.0	**75.6**	3428.7
	mafftmerge (default)	N.A.	**64.9**	76.7	70.8	**10.4**
tRNA-synt_2b (11,293)	gcm (default)	12,936,532	48.8	**77.8**	**63.3**	1.1
	gcm (fm+opt)	**13,236,030**	**52.0**	61.1	56.6	38.0
	mafftmerge (default)	N.A.	43.4	53.7	48.6	**0.3**
zf-CCHH (88,345)	gcm (default)	32,422,598	79.9	**93.8**	86.8	**3.8**
	gcm (fm+opt)	**32,551,838**	**86.8**	89.4	**88.1**	51.9
	mafftmerge (default)	N.A.	NaN	NaN	NaN	NaN
Average	gcm (default)	22,542,507	68.0	**90.5**	**79.3**	78.0
	gcm (fm+opt)	**23,179,541**	**71.9**	82.5	77.2	358.7
	mafftmerge (default)	N.A.	68.7	77.9	73.3	**1.7**

forms relatively well, generally as accurate as GCM. However, MAFFT–merge with L-ins-i is more computationally intensive than GCM, making GCM more appealing on large datasets. Furthermore, we note that we used MAFFT–merge in a way that is not recommended (see the MAFFT–merge website at https://mafft.cbrc.jp/alignment/software/merge.html, which recommends that MAFFT–merge use monophyletic clusters as entries). Although monophyly is not ensured in PASTA decompositions, it is definitely violated in random decompositions, showing that MAFFT–merge (using L-ins-i) is a valuable approach for merging non-monophyletic clusters, even though it wasn't designed for it initially.

5 Conclusions

This study was motivated by the desire to understand why MAGUS, a recent method for multiple sequence alignment, produced more accurate alignments than its immediate predecessor, PASTA. Both MAGUS and PASTA operate by dividing a sequence dataset into disjoint sets, compute alignments on the subsets using MAFFT -linsi, and then merge these disjoint alignments together; essentially the only important difference between the two methods is how each merges the disjoint alignments. MAGUS computed a set of extra alignments in order to merge the disjoint alignments, defined an edge-weighted "alignment graph" from these extra alignments, and then computed a "trace" (i.e., merged alignment) using the Graph Clustering Merger (GCM) applied to the alignment graph. In contrast, to merge the set of disjoint alignments, PASTA used a much simpler strategy: it merged pairs of the constraint alignments and then followed this by transitivity. Thus, GCM uses a more complex approach than PASTA's merger strategy, but neither is based on an explicit optimization criterion. In exploring the GCM method further, we hypothesized that its approach might be an effective technique with respect to maximizing the support implied by the alignment graph, which we defined to be the total weight of the edges in the alignment graph that appear in the output alignment. We formulated this objective as a new optimization problem, which we termed the Maximum Weight Trace Alignment Merging (MWT-AM) problem. Thus, the MWT-AM problem is a generalization of a classical problem in bioinformatics called the Maximum Weight Trace problem [12]. Our study shows that the Graph Clustering Merger (GCM) method provides good accuracy for the MWT-AM problem. Further, our study shows that using default GCM or its improved variant, GCM(FM+opt), within the MAGUS pipeline produces alignments with excellent accuracy, and can be used on very large datasets (up to 93,681 sequences in this study).

This study suggests multiple directions for future research. The first and most obvious direction is to develop new methods for merging a set of disjoint alignments. Currently, there are only a few methods that are able to merge disjoint alignments, and so there is clear opportunity for advances. To begin with, future research could explore modifying how GCM defines the weights on the pairs of columns, as the heuristic it uses to solve MWT-AM might work well with other techniques. Also of interest is the theoretical approximability of MWT and MWT-AM; to the best of our knowledge, there are no results regarding polynomial-time constant-factor approximations for these problems.

Given the high accuracy of MAGUS, which depends on GCM, another direction for future work is to investigate changes to the ways the constraint alignments and backbone alignments are computed within MAGUS. In this study we relied on MAFFT for these computations, and MAFFT has been shown to generally provide very good results for alignment estimation, especially when used in its most accurate settings [1], [2], [6], [7]. Yet other methods could also be considered and could be computationally feasible. Based on results shown in [27], when used with methods for aligning amino acid sequences, such as PROMALS [28] and ProbCons [29], the MAGUS pipeline could potentially be even more accurate. Another interesting direction is to consider the use of computationally intensive Bayesian methods for the constraint alignments. For example, methods such as BAli-Phy [30] can have outstanding accuracy, but are generally limited to very small datasets (perhaps 50 sequences). As noted in [31], BAli-Phy can be used within the PASTA divide-and-conquer strategy, where it is only used on small datasets; similarly, we conjecture that the MAGUS pipeline could be used with BAli-Phy on subsets to improve accuracy.

Finally, alignment estimation is used in many applications, including computational linguistics [32], [33], [34], [35], [36], [37], [38], [39], [40], speech processing [41], [42], [43], biological networks [44], [45], [46], [47], and problems involving process models [48], [49], [50], [51]. It is likely that techniques and approaches for alignment estimation in these and other disciplines may be relevant to the MWT-AM problem we present here, and future work should investigate this.

Appendix

Data availability

All datasets studied in this paper are from prior publications, and are available online:

- RNASim2: available at https://doi.org/10.5061/dryad.95x69p8h8 and https://databank.illinois.edu/datasets/IDB-6955387
- HomFam datasets available at https://databank.illinois.edu/datasets/IDB-1048258
- Everything else: available at https://databank.illinois.edu/datasets/IDB-2643961

Decomposition into subsets with PASTA

We used the set of subsets decomposed by PASTA during the first iteration, using the following command:

```
python run_pasta.py -i <set of unaligned
sequences> --keeptemp --keepalignmenttemps
--temporaries=<temporary folder>
```

By default the maximum subset size is set to `200`.

MAFFT v7.453 (constraint alignments)

Mafft was run on constraint and backbone alignments using the `l-ins-i` algorithm:

```
mafft --localpair --maxiterate 1000 --ep
0.123 --quiet --thread <available number of
threads> --anysymbol input > results
```

MAFFT v7.453 (merger) using l-ins-i

```
mafft --localpair --maxiterate 1000 --ep
0.123 --quiet --anysymbol --merge <list of
constraint alignments> > result
```

MAFFT v7.453 (merger) using default

```
mafft --merge <list of constraint
alignments> > result
```

T-Coffee 13.41.0

```
t-coffee -profile <list of constraint
alignments> -output fasta -outfile
result.txt
```

GCM (within MAGUS)

To run GCM, download and install MAGUS from the github site https://github.com/vlasmirnov/MAGUS, and then use the commands provided here:

```
   python3 magus.py -d <working directory>
-s <list of constraint alignments> -r
<number of backbone alignments> -m <backbone
alignment size> -o result.txt
```

GCM: additional flags in the magus.py script

Clustering:

```
--graphclustermethod mcl|mlrmcl|rg|none
```

Trace:

```
--graphtracemethod
minclusters|fm|mwtgreedy|mwtsearch|rg|rgfast
```

Optimizer:

```
--graphtraceoptimize true|false
```

Acknowledgments

This research was supported by the US National Science Foundation grant 2006069 and ABI: 1458652 (to TW) and the Debra and Ira Cohen Fellowship (to VS).

References

[1] K. Liu, S. Raghavan, S. Nelesen, C. R. Linder, and T. Warnow, "Rapid and accurate large-scale coestimation of sequence alignments and phylogenetic trees," *Science*, vol. 324, no. 5934, pp. 1561–1564, 2009.

[2] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, "PASTA: ultra-large multiple sequence alignment for nucleotide and amino-acid sequences," *Journal of Computational Biology*, vol. 22, no. 5, pp. 377–386, 2015.

[3] N.-p. D. Nguyen, S. Mirarab, K. Kumar, and T. Warnow, "Ultra-large alignments using phylogeny-aware profiles," *Genome Biology*, vol. 16, no. 1, p. 124, 2015.

[4] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins, "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega," *Molecular Systems Biology*, vol. 7, no. 1, p. 539, 2011.

[5] R. Hagopian, J. R. Davidson, R. S. Datta, B. Samad, G. R. Jarvis, and K. Sjolander, "SATCHMO-JS: a webserver for simultaneous protein multiple sequence alignment and phylogenetic tree construction," *Nucleic Acids Research*, vol. 38, no. suppl_2, pp. W29–W34, 2010.

[6] K. Liu, T. J. Warnow, M. T. Holder, S. M. Nelesen, J. Yu, A. P. Stamatakis, and C. R. Linder, "SATe-II: very fast and accurate simultaneous estimation of multiple sequence alignments and phylogenetic trees," *Systematic biology*, vol. 61, no. 1, p. 90, 2012.

[7] V. Smirnov and T. Warnow, "MAGUS: multiple sequence alignment using graph clustering," *Bioinformatics*, vol. 37, no. 12, pp. 1666–1672, 2021.

[8] T. J. Wheeler and J. D. Kececioglu, "Multiple alignment by aligning alignments," *Bioinformatics*, vol. 23, no. 13, pp. i559–i568, 2007.

[9] R. C. Edgar, "MUSCLE: a multiple sequence alignment method with reduced time and space complexity," *BMC bioinformatics*, vol. 5, no. 1, p. 113, 2004.

[10] J. Kececioglu, "The maximum weight trace problem in multiple sequence alignment," in *Annual Symposium on Combinatorial Pattern Matching*. Springer, 1993, pp. 106–119.

[11] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J. D. Kececioglu, "A branch-and-cut algorithm for multiple sequence alignment," in *Proceedings of the First Annual International Conference on Computational Molecular Biology*, 1997, pp. 241–250.

[12] J. D. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron, "A polyhedral approach to sequence alignment problems," *Discrete applied mathematics*, vol. 104, no. 1-3, pp. 143–186, 2000.

[13] G. Koller and G. R. Raidl, "An evolutionary algorithm for the maximum weight trace formulation of the multiple sequence alignment problem," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2004, pp. 302–311.

[14] E. Moreno-Centeno and R. M. Karp, "The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment," *Operations Res*, vol. 61, no. 2, pp. 453–468, 2013.

[15] K. Katoh, K.-i. Kuma, H. Toh, and T. Miyata, "MAFFT version 5: improvement in accuracy of multiple sequence alignment," *Nucleic Acids Research*, vol. 33, no. 2, pp. 511–518, 2005.

[16] C. Notredame, D. G. Higgins, and J. Heringa, "T-Coffee: A novel method for fast and accurate multiple sequence alignment," *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[17] S. M. Van Dongen, "A cluster algorithm for graphs," National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, Tech. Rep., May 2000, iNS-R0010.

[18] V. Satuluri and S. Parthasarathy, "Scalable graph clustering using stochastic flows: applications to community discovery," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 737–746.

[19] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[20] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *19th design automation conference*. IEEE, 1982, pp. 175–181.

[21] M. Modzelewski and N. Dojer, "MSARC: multiple sequence alignment by residue clustering," *Alg Mol Biol*, vol. 9, no. 1, p. 12, 2014.

[22] J. J. Cannone, S. Subramanian, M. N. Schnare, J. R. Collett, L. M. D'Souza, Y. Du, B. Feng, N. Lin, L. V. Madabusi, K. M. Müller, N. Pande, Z. Shang, N. Yu, and R. R. Gutell, "The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs," *BMC Bioinf*, vol. 3, no. 1, p. 2, 2002.

[23] F. Sievers, D. Dineen, A. Wilm, and D. G. Higgins, "Making automated multiple alignments of very large numbers of protein sequences," *Bioinformatics*, vol. 29, no. 8, pp. 989–995, 2013.

[24] S. Mirarab and T. Warnow, "FastSP: linear time calculation of alignment accuracy," *Bioinformatics*, vol. 27, no. 23, pp. 3250–3258, 2011.

[25] J. Stoye, D. Evers, and F. Meyer, "Rose: generating sequence families," *Bioinf*, vol. 14, no. 2, pp. 157–163, 1998.

[26] V. Smirnov, " Recursive MAGUS: scalable and accurate multiple sequence alignment," *PLoS Computational Biology*, vol. 17, no. 10, p. e1008950, 2021.

[27] M. Nute, E. Saleh, and T. Warnow, "Evaluating statistical multiple sequence alignment in comparison to other alignment methods on protein data sets," *Systematic biology*, vol. 68, no. 3, pp. 396–411, 2019.

[28] J. Pei and N. V. Grishin, "PROMALS: towards accurate multiple sequence alignments of distantly related proteins," *Bioinformatics*, vol. 23, no. 7, pp. 802–808, 2007.

[29] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "Probcons: Probabilistic consistency-based multiple sequence alignment," *Genome research*, vol. 15, no. 2, pp. 330–340, 2005.

[30] M. A. Suchard and B. D. Redelings, "BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny," *Bioinformatics*, vol. 22, no. 16, pp. 2047–2048, 2006.

[31] M. Nute and T. Warnow, "Scaling statistical multiple sequence alignment to large datasets," *BMC genomics*, vol. 17, no. 10, pp. 135–144, 2016.

[32] J. Eisner, "Learning non-isomorphic tree mappings for machine translation," in *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*, 2003, pp. 205–208.

[33] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[34] I. D. Melamed, "Bitext maps and alignment via pattern recognition," *Computational Linguistics*, vol. 25, no. 1, pp. 107–130, 1999.

[35] G. Jäger, "Support for linguistic macrofamilies from weighted sequence alignment," *Proceedings of the National Academy of Sciences*, vol. 112, no. 41, pp. 12 752–12 757, 2015.

[36] A. Fraser and D. Marcu, "Measuring word alignment quality for statistical machine translation," *Computational Linguistics*, vol. 33, no. 3, pp. 293–303, 2007.

[37] Y. Liu, Q. Liu, and S. Lin, "Tree-to-string alignment template for statistical machine translation," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 2006, pp. 609–616.

[38] M. A. Covington, "Alignment of multiple languages for historical comparison," in *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

[39] J. Tiedemann, "Lingua-align: An experimental toolbox for automatic tree-to-tree alignment," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, 2010.

[40] U. D. Reichel, "PermA and Balloon: Tools for string alignment and text processing," in *Proc. Interspeech*, 2012.

[41] C. Cerisara, O. Mella, and D. Fohr, "JTrans, an open-source software for semi-automatic text-to-speech alignment," in *Proceedings of the 10th Annual Conference of the International Speech Communication Association-Interspeech 2009*, 2009.

[42] Y.-A. Chung, W.-H. Weng, S. Tong, and J. Glass, "Unsupervised cross-modal alignment of speech and text embedding spaces," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[43] A. Katsamanis, M. Black, P. G. Georgiou, L. Goldstein, and S. Narayanan, "Sailalign: Robust long speech-text alignment," in *Proc. of Workshop on New Tools and Methods for Very-Large Scale Phonetics Research*, 2011.

[44] R. Patro and C. Kingsford, "Global network alignment using multiscale spectral signatures," *Bioinformatics*, vol. 28, no. 23, pp. 3105–3114, 2012.

[45] S. Zhang and H. Tong, "Network alignment: recent advances and future directions," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 3521–3522.

[46] B. P. Kelley, R. Sharan, R. M. Karp, T. Sittler, D. E. Root, B. R. Stockwell, and T. Ideker, "Conserved pathways within bacteria and yeast as revealed by global protein network alignment," *Proceedings of the National Academy of Sciences*, vol. 100, no. 20, pp. 11 394–11 399, 2003.

[47] R. Sharan and T. Ideker, "Modeling cellular machinery through biological network comparison," *Nature biotechnology*, vol. 24, no. 4, pp. 427–433, 2006.

[48] G. Bergami, F. M. Maggi, A. Marrella, and M. Montali, "Aligning data-aware declarative process models and event logs," in *International Conference on Business Process Management*. Springer, 2021, pp. 235–251.

[49] G. Bergami, F. M. Maggi, M. Montali, and R. Peñaloza, "Probabilistic trace alignment," in *2021 3rd International Conference on Process Mining (ICPM)*. IEEE, 2021, pp. 9–16.

[50] A. Adriansyah, B. F. van Dongen, and W. M. van der Aalst, "Conformance checking using cost-based fitness analysis," in *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*. IEEE, 2011, pp. 55–64.

[51] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance checking: Relating processes and models*. Switzerland: Springer, 2018.

Paul Zaharias is a postdoctoral researcher at Muséum national d'Histoire naturelle de Paris working with Olivier Gascuel and former postdoctoral researcher (with Tandy Warnow) at the University of Illinois Urbana-Champaign. He received his PhD in 2019 from the Muséum national d'Histoire naturelle de Paris under the direction of Nicolas Puillandre, where he worked on the systematics and evolution of marine molluscs. His current research interests include metrics for support in phylogenies, computational methods for large-scale multiple sequence alignment and phylogeny estimation in the presence of substantial heterogeneity across the tree and across the genome.

Vladimir Smirnov is a postdoctoral researcher at the University of Cambridge, working with Richard Durbin. He received his PhD in 2021 from the University of Illinois Urbana-Champaign, under the direction of Tandy Warnow. His main research interests are in large-scale multiple sequence alignment and related bioinformatics problems. His awards include the Gene Golub Fellowship (2018) and the Debra and Ira Cohen Fellowship (2021).

Tandy Warnow is the Grainger Distinguished Chair in Engineering in the Department of Computer Science at the University of Illinois Urbana-Champaign. Tandy received her PhD in Mathematics at UC Berkeley in 1991 under the direction of Gene Lawler, and her research focuses on reconstructing complex and large-scale evolutionary histories. She was awarded the David and Lucile Packard Foundation Award (1996), a Radcliffe Institute Fellowship (2003), and the John Simon Guggenheim Foundation Fellowship (2011). She was elected Fellow of the Association for Computing Machinery (ACM) in 2015, Fellow of the International Society for Computational Biology in 2017, and Fellow of the Association for the Advancement of Science (AAAS) in 2021.