Multiple Sequence Alignment: A Critical Comparison of Four Popular Programs

Shirley Sutton, Biochemistry 218 Final Project, March 14, 2008

Introduction

For both the computational biologist and the research biologist, the use of multiple sequence alignment (MSA) programs to simultaneously align multiple sequences of nucleic acids or proteins has become *de rigueur*. Successful alignments are used in a number of applications, such as (1) phylogenetic analysis, as a predictor of evolutionary relationships; (2) identifying conserved motifs and domains within related families of proteins that then may be inferred to play a role in structure and function; (3) structure prediction, in which the role of a residue in secondary or tertiary structure (e.g., α helix, β sheet, etc.,) is inferred.

Since the first MSA programs became commonly available in the late 1980's, a number of "new and improved" programs have been introduced, giving the researcher a wide variety of choices. Because the many MSA programs use a variety of methods and can yield different results, it can be difficult for the researcher to know which program to choose. When asked what the important considerations are in constructing a MSA, most researchers will say that they want a program that provides speed and accuracy. However, even these two simple parameters can have very different meanings to researchers in different fields. The performance speed that is acceptable to a research biologist might prove to be prohibitively slow to a computational biologist. Similarly, "accuracy" will depend on several factors related to the types of sequences being aligned--whether it is nucleic acid or protein families that are being compared, how related the families are, what sequence lengths are being compared, how many sequences are being compared, etc.--as well as to the methods employed by the MSA algorithm--progressive vs. iterative, matrix-based vs. consistency-based, etc. When faced with such an array of considerations, researchers will often simply choose a program with which they are most familiar.

In this paper, we review and analyze some of the more popular and available MSA programs: CLUSTAL W, T-Coffee, PROBCONS and MUSCLE. We begin by reviewing the general methodologies behind MSA programs, followed by a summary of the four programs listed above. We finish with a comparison and analysis of the presented programs.

The Methodology

The purpose of an MSA is to align sequences in such a way as to reflect the biological relationship between the input sequences, but developing a reliable MSA program is a very complex problem. In its most basic form the problem can be stated in the following way: given N sequences and a scoring scheme for determining the best matches of the letters (where each sequence consists of a series of letters), find the

optimal pairing of letters between the sequences. Even this simplistic definition requires a consideration of the choice of sequence, choice of comparison model and optimization of the model.

The most common algorithms in use today are progressive alignments, based on the work of Feng and Doolittle (1987). These algorithms are by nature heuristic and therefore do not guarantee any level of optimization, but they do tend to be fast (Notredame, 2002). In brief, a set of N sequences are aligned by performing N-1 pairwise alignments of pairs of proteins or pairs of intermediate alignments to create a distance matrix. The results of the distance matrix are used to create a (phylogenetic) guide tree. Sequences are added to the alignment one by one, with the most closely related sequences aligned first, followed by the more distantly related ones, inserting gaps, if necessary (Edgar and Batzoglou, 2006; Notredame, 2007).

The most influential component of progressive algorithms is the scoring schemes used by pairwise alignments (Notredame, 2007), in which the best alignment of two sequences is defined as the sum of substitution matrix scores for each pair of letters, minus gap penalties. Note that even a pairwise sequence alignment must be further categorized as being either a global alignment--one that spans the entire length of the input sequence--or a local alignment--regions of aligned sub-sequences that may be surrounded by sequences that are completely unrelated.

Pairwise alignment schemes can be divided into two types: matrix-based and consistency-based. Matrix-based algorithms use a substitution matrix to determine the cost of matching two letters. The important point here is that the score for matching two letters depends only on their position and their immediate surroundings. Consistency-based algorithms utilize a much larger volume of information: pairwise global and local alignments are compiled; these compiled alignments are used as a position-specific substitution matrix during a normal progressive alignment. As an example of consistency-based scoring, consider three sequences A, B, and C. The pairwise alignment of A-B and B-C may produce an alignment of A and C that is different from a directly computed A-C alignment and may give us information as to the "correct" alignment between A and C (Edgar and Batzoglou, 2006).

Iterative alignments are a refinement of progressive alignments. Instead of depending heavily on the accuracy of the initial pairwise alignments, as the progressive schemes do, iterative alignments optimize an objective function: they use an algorithm that produces an alignment that is then refined through a series of cycles until no more improvements can be made (Notredame, 2002). There are a variety of ways to select the sequence subgroups and the objective function (see, for example, Hirosawa et. al., 1995). Depending on the strategy used to improve the alignment, the iterative method can be either stochastic or deterministic. Deterministic strategies are the simpler of the two. Sequences are pulled one by one from an alignment and then re-aligned to the remaining sequences. When no more improvement can be made (i.e., convergence), the process is halted. Stochastic iterative methods use HMM training and simulated annealing or genetic algorithms.

The latest generation of MSA programs incorporates constraint-based methods into their progressive algorithms. The constraints generally come from three different

sources: (1) the use of biologically relevant information encoded in databases such as PROSITE; (2) the use of local pairwise similarity present in multiple sequence pairs to highlight similar regions in otherwise divergent sequences; (3) user-specified regions in sequences they wish to see aligned in any multiple alignment computed (Papadopolous et. al., 2007).

The Programs

CLUSTAL W

The CLUSTAL set of MSA programs were first developed in 1988. As stated by Higgins and Sharp in their 1988 paper, CLUSTAL is essentially a "quick and dirty" version of the Feng and Doolittle (1987) pairwise progressive algorithm. What made the CLUSTAL programs so attractive is that they could be run on so-called microcomputers and therefore any researcher with a computer could perform sequence alignments right in the lab.

Many of the algorithms developed after CLUSTAL are at least in part based on the CLUSTAL model, so it is worth exploring this algorithm in some detail.

There are three separate and distinct steps in the CLUSTAL MSA (Figure 1):

- Calculation of all pairwise sequence similarities, which are then used to construct a distance matrix (also called a similarity matrix)
- Construction of a dendogram (guide tree) from the distance matrix generated in step 1
- Alignment of the sequences in a pairwise manner, following the order of the clustering as determined by the guide tree generated in step 2

CLUSTAL W, an improvement on the original CLUSTAL program was introduced in 1994 (Thompson et. al., 1994), and its relative speed and sensitivity soon made it the method of choice for biologists. The biggest problem with the original CLUSTAL programs is that their algorithms made certain assumptions that are not biologically realistic. The enhanced sensitivity was due to three improvements incorporated into CLUSTAL W: the use of a weighted sum-of-pairs, the use of varying gap penalties and the use of neighbor-joining instead of UPGMA in the generation of the phylogenetic tree.

In older alignment algorithms, single-weight matrices were generated from the pairwise alignments. A single-weight matrix assumes that the sequences in a group to be aligned are all equally divergent from each other. As long as this assumption is true then the use of single weight matrices is justified. However, if the sequences in an alignment group are too similar, then a single-weight matrix introduces a bias in favor of redundant sequences. If the sequences are divergent, then mismatches in sequence become more prevalent but single-weight matrices do not account for them. CLUSTAL W assigns individual weights to sequences, in part by using neighbor joining to construct the phylogenetic tree: weights are assigned according to the tree branch length, which is a measure of their evolutionary distance. This means that similar or

redundant sequences will be downweighted while more divergent sequences are upweighted.

CLUSTAL W also changed the way that gap penalties and gap opening were determined. To get an alignment between divergent sequences, alignment algorithms use gap penalties and gap extensions, but the older algorithms assigned fixed values to both. This is not biologically realistic, as the gaps found in related proteins are not random occurrences. Regions of conserved structure or catalytic function (e.g., the motor domains of myosin or the P-loop nucleotide binding regions found in many ATPases) are much less likely to have gaps than the linkers that connect these structure/function domains. Before any pair of sequences or prealigned groups of sequences are aligned, CLUSTAL W generates a table of gap opening penalties for every position in the two (sets of) sequences. The initial gap-opening penalty is varied in both a residue and a position-specific manner, in order to make gaps more or less likely at different positions. The residue-specific gap penalties and locally reduced gap penalties in hydrophilic regions encourage new gaps in potential loop regions rather than regular secondary structure. Finally, positions in early alignments where gaps have been opened receive locally reduced gap penalties to encourage the opening up of new gaps at these positions.

Although there have been no substantive changes to CLUSTAL W since it was first released in 1994, a new member of the CLUSTAL family, CLUSTAL X, was introduced in 1997 (Thompson, et. al., 1997). This version uses the same algorithm as CLUSTAL W, but it has features that make it more user friendly (e.g., scrollable windows, pull-menus, etc.) For a review of the CLUSTAL programs, see Chenna, et. al. (2003).

T-Coffee

In 2000 (Notredame et. al., 2000), the T-Coffee (Tree-based Consistency Objective Function for alignment Evaluation) alignment program was introduced. This was the first alignment program to introduce any meaningful improvements on the CLUSTAL W algorithm. Although T-Coffee is still a progressive alignment method, it is consistency-based and so takes advantage of a much larger body of information. It also has the ability to consider information from all of the sequences during each alignment step and not just those sequences being aligned at a certain stage.

Programs such as CLUSTAL W are called "greedy heuristics." As we have previously described them, they start with pairwise alignments of sequences and generate a distance matrix. This matrix is used to generate a phylogenetic tree and then the tree is used to gradually build up the alignment by following the branches of the tree. This method has proved successful for a number of applications, but it does possess a flaw that we have not yet discussed: if an error is made in the first set of alignments, this error is then propagated through the rest of the alignments and cannot be rectified later as the rest of the sequences are added in. Although T-Coffee is itself a greedy heuristic, it attempts to minimize the effect caused by errors in alignment that happen early on, by making better use of the available information.

T-Coffee has two main features. First, it uses heterogeneous data sources to generate multiple alignments. The data from these sources are provided to T-Coffee via a library of pairwise alignments. Thus, T-Coffee can compute MSAs using a library that was generated from a mixture of local and global pairwise alignments. Second, it carries out progressive alignment in a way that allows it to consider the alignment between all of the pairs during the generation of the MSA. This gives it the speed of a traditional progressive alignment but with far less tendency to misalign.

The basic steps in a T-Coffee MSA are as follows (see Figure 2 for a schematic):

- Generate a primary library of alignments
  - Contains a set of pairwise alignments between all of the sequences to be aligned
  - Two alignment sources--one local, one global--for each pair of sequences are used; yielding two libraries--one local, one global
- Derive the weights for the primary library
  - A weight is assigned to each pair of aligned residues in the library
  - The weights are assigned according to sequence identity
- Combination of the libraries
  - The libraries are pooled in a simple addition process
  - Duplicated pairs are merged into a single entry
  - Pairs that did not occur will not be represented (given a weight of zero)
- Extending the library
  - Combine information so that the final weight, for any pair of residues, reflects some of the information contained in the whole library
  - Based on taking each aligned pair from the library and checking the alignment of the two residues with residues from the remaining sequences (consistency-based alignment)
- Progressive alignment
  - Score for the alignments x_i to y_j is the sum of the weights of the alignments in the library containing the alignment x_i to y_j.
  - The distance matrix and neighbor-joining tree are determined'
  - The initial pair is fixed at this point; any existing gaps cannot be shifted later

MUSCLE

In 2004 the MUSCLE (multiple sequence comparison by log-expectation) algorithm was introduced (Edgar, 2004). MUSCLE is a matrix-based algorithm, and like other MSA algorithms, MUSCLE starts with a guide tree construction, where the fundamental step is pairwise alignment. The pairwise alignment is used first for progressive alignment and then for refinement. There are two distinguishing feature of MUSCLE. In determining distance measures for pairs of sequences, it uses both the kmer distance (for an unaligned pair) and the Kimura distance (for an aligned pair). A kmer is merely a contiguous sequence of letters of length k, also known as a word or k-tuple. Sequences that are related will have more kmers in common than expected by

chance. The kmer distance is derived from the fraction of kmers in common in a compressed alphabet, which is related to fractional identity. Because this measure does not require an alignment, MUSCLE has a significant speed advantage over other MSA algorithms. The second distinguishing feature is that at the completion of any stage of the algorithm, a MSA is available and the algorithm can be terminated.

Interestingly, distance matrices in MUSCLE are clustered using UPGMA, which the author says give marginally improved results over neighbor-joining, even though neighbor-joining gives a more reliable estimate of the taxonomic evolutionary tree. He explains this by saying that in progressive alignment, the best accuracy is obtained at each node by aligning the two profiles that have fewest differences, even if those profiles are not evolutionary neighbors.

There are three main stages to the MUSCLE algorithm, as shown in Figure 3. For stage 1, a progressive alignment of the sequences is generated. The kmer distance for each pair of input sequences is calculated, and from this a matrix is constructed. The matrix is used to construct the UPGMA tree. From the tree, the progressive alignment is constructed by following the branching order of the tree. At each leaf, a profile is constructed from an input sequence. Nodes in the tree are visited in prefix order (children before their parent). At each internal node, a pairwise alignment is constructed of the two child profiles, giving a new profile that is assigned to that node. This produces an MSA.

Stage 1 introduces error in the form of the approximate kmer distance, which produces a suboptimal tree. Stage 2, the improved progression stage, improves on the alignment generated in stage 1. MUSCLE starts by re-estimating the tree using the Kimura distance, which is more accurate but requires an alignment. A progressive alignment is produced, yielding a second MSA. The trees from stages 1 and 2 are compared; MUSCLE identifies a set of nodes for which the branching order is different. A new MSA is built if the order of the nodes has changed. Otherwise the first MA is kept.

Stage 3 is the refinement stage. Here, an edge of the tree from stage 2 is deleted. This divides the tree into two sub-trees; the profile of the multiple alignment for each sub-tree is calculated. The profiles from the two sub-trees are re-aligned, producing a new MSA. If the new sum-of-pairs score is improved, the new alignment is kept. Otherwise it is discarded. These steps are repeated until convergence or until a user-defined limit is achieved.

ProbCons

The last algorithm we will present is ProbCons--probabilistic consistency-based MSA (Do et. al., 2005). It is a progressive alignment consistency-based algorithm, but ProbCons takes a very different approach to the formulation of the sequence alignment problem: it uses a three-state pair-hidden Markov model (HMM) as an alternative formulation of the sequence alignment problem (Figure 4, Do et. al, 2005) where emissions correspond to traditional substitution scores based on the BLOSUM62 matrix and transitions correspond to gap penalties which are trained with unsupervised expectation maximization.

Other features that distinguish ProbCons from the algorithms we have presented thus far include its use of maximum efficiency accuracy instead of Viterbi alignment, which is more commonly used on HMMs to generate sequence alignment. A Viterbi alignment is similar to that of Needleman-Wunch, in that it selects a single alignment with the highest probability of being absolutely correct. In contrast, maximum expected accuracy selects the alignment with the largest number of correct predictions. ProbCons uses probabilistic consistency transformation to incorporate multiple sequence conversion information during pairwise alignment. This is a modification of the sum-of-scores method: the transformation is to re-estimate the posterior probabilities using three-sequence alignments instead of pairwise alignments.

One final feature of note: ProbCons does not use any biological concepts such as evolutionary guide tree construction and position-specific gap scoring in its algorithm.

The ProbCon algorithm has five main steps:

- Computation of posterior-probabilities matrices
  - For every pair of sequences x and y, a matrix is computed where the terms of the matrix are the probabilities that letter x_i and y_j are paired in an alignment of x and y as generated by the model
- Computation of expected accuracies
  - The expected accuracy of a pairwise alignment *a* between x and y to be the expected number of correctly aligned pairs of letters, divided by the length of the shorter sequence
- Probabilistic consistency transformation
  - Re-estimate the matrix quality scores by applying the probabilistic consistency transformation
- Computation of a guide tree
  - Use hierarchical clustering
- Compute progressive alignment
  - Align sequence groups according to order specified in the guide tree

A post-processing iterative step can be applied as necessary. Here, the alignment is randomly partitioned into two groups of sequences and re-aligned as required.

DALIGN and COBALT

In the process of preparing this paper, we learned of two other algorithms that deserve at least an honorable mention. The distinguishing feature of DALIGN (Morgenstern, 1998) is that it aligns sequences both locally and globally using a diagonal method (hence the name DALIGN). Rather than compare single residues, it compares whole interrupted stretches of residues that would form diagonals in a dot matrix; it does not allow for mismatches or gaps. As a result, it has no gap penalty or gap extension, and may leave unrelated sequences unaligned.

COBALT (constraint-based alignment tool) is one of the latest algorithms to be released (Papadopoulos and Agarwala, 2007). What makes COBALT unique is that it allows the user to enter constraints: the user can directly specify pairwise constraints

and/or can ask COBALT to generate the constraints using sequence similarity, (optional) CDD (conserved domain database) searches and (optional) PROSITE pattern searches. COBALT will optionally create partial profiles for input sequences based on any CDD search result.

Analysis

When assessing the performance of a MSA algorithm, the accepted way of doing this is to do empirical tests using an established database of test sequences. To avoid potential pitfalls that are inherent in this method of testing (some algorithms could be over-trained on a specific test by finding the parameter combinations that make a package look best with a particular set of test cases; global aligners tend to over-align sequences by aligning residues and regions of the sequences outside of the conserved core that share no structural or homologous relationship; local aligners tend to misalign), many software developers now test their programs on several of the many benchmarking suites now available, such as BAliBASE, PREFAB and SABmark. Recently, a new simulation software package, Simprot, has been introduced (Nuin et. al., 2006). How these alignment databases are generated and what methods they use to determine an accuracy score is a topic for another paper; we will assume that they have been tested and found to be useful tools. (For an analysis of the currently available benchmarks, see Blackshields et. al. 2006).

Tables 1, 2 and 3 show the results of testing MSA algorithms with BAliBASE, PREFAB and SABmark, respectively. Note that the tests were run on several programs besides just the four that were reviewed in this paper. From these results we see certain trends. Consider first an overall score for accuracy. In all cases, a higher overall score indicates a better performance. Of the four programs reviewed here, ProbCons consistently has the highest overall score, outperforming CLUSTAL W, T-Coffee and MUSCLE on all three benchmarks. This is particularly significant because ProbCons was trained on BAliBASE (Do et. al., 2005), so its performance on PREFAB and SABmark provide external validations of the BAliBASE results. CLUSTAL W, on the other hand, had the lowest overall score of the four programs that we reviewed (although not the lowest score of all of the programs tested; DALIGN scored even lower than CLUSTAL W on all three benchmarks). Note that CLUSTAL W scored better on BAliBASE than it did on the other two benchmarks. CLUSTAL W was trained on BAliBASE, so it is possible that the higher score with this benchmark reflects over-training. T-Coffee and MUSCLE are intermediate in accuracy, falling between ProbCons and CLUSTAL W.

If we now consider running time, MUSCLE outperforms all of the programs reviewed here, in some cases by as much as 20 times (for example, compare the run times of MUSCLE and T-Coffee when the BAliBASE benchmark was used, Table 1). MUSCLE was optimized for speed (Edgar 2004), so its performance speed is expected to be high. Contrast this with T-Coffee, the slowest of the four programs. It is interesting to note that the speed of T-Coffee is, in all cases, at least an order of magnitude slower than MUSCLE, the fastest program, although its accuracy as determined by its overall score is comparable to both CLUSTAL W and MUSCLE.

The four programs reviewed here have also been tested with Simprot, simulation software that generates known alignments under realistic and controlled evolutionary scenarios (Nuin et. a., 2006). For these tests, simulated sequences were used to investigate the effects of sequence length, indel frequency and length, evolutionary distance, terminal gap length and tree topology. Figure 5 shows the overall results of these tests and compares them to the results from a BAliBASE test on the same programs. ProbCons generated the best alignments with Simprot, while CLUSTAL W (and DALIGN) had the worst accuracy. T-Coffee and MUSCLE fall somewhere in the middle. These results follow the trend seen with BAliBASE, PREFAB and SABmark.

For completeness sake, we also report on how COBLAT compares to the four programs reviewed in this paper. Figure 4 shows that COBALT outperforms ProbCons on both the BAliBASE and PREFAB benchmarks but not on SABmark; COBLAT does better than MUSCLE when both programs are tested with SABmark.

Finally, we mention a report that tests the programs on their ability to detect and align multiple sequences that possess insertions and deletions. These are common features in biologically relevant sequences, and their presence does affect the accuracy of MSA packages, but the extent to which alignment accuracy is affected by the position of insertions and deletions is not often examined independently of other sources of sequence variation (Golubchik et. al., 2007). In brief, data sets were constructed from sequences chosen on the basis of their different lengths and organisms of origin, so as to control for sequence-specific effects in the alignment. For each sequence so chosen, it was replicated 10 times (Figure 6). A stretch of either 10 or 30 residues was deleted from each sequence. This shifted the gap origin either by one residue for overlapping deletions or by the length of the deletion for no-overlapping deletions. MSA programs were then assessed for their ability to correctly place overlapping gaps within sequences that contained overlapping deletions. The results are shown in Figure 7. All four programs reviewed here preferentially aligned gaps in a single vertical column rather than the expected diagonally staggered band. Different programs opened the gap at different positions. DALIGN was the only exception, being able to recreate the correct staggered arrangement in over 60% of the tested alignments.

Conclusions

Speed, accuracy, memory: these are the three main criteria on which all MSA algorithms are judged. Of the three, biological accuracy is arguably the most important. With the many MSA algorithms now available, and with their increasingly similar performances and accuracy, it is difficult to objectively choose one method over another. Improvements to both precision and speed are continuously being enacted. Nevertheless, analysis of some popular algorithms show that, depending on what is important to the researcher, one method may be preferred over another.

CLUSTAL W, the oldest of the programs reviewed here, showed respectable performance in both speed and accuracy, although there are certainly programs that are faster and more accurate. The advantage of CLUSTAL W is that it strikes a balance between these

measures. Disadvantages of CLUSTAL W are that it has no objective function and there is no real way to quantify the resulting alignment.

T-Coffee offers a distinct accuracy improvement over CLUSTAL W, by using a combination of local and global pairwise alignments to generate the sequence library, but its incredibly slow speed may keep it from being widely implemented.

MUSCLE, with its unique way of calculating distance measures (using kmer distance for an unaligned pair and Kimura distance for an aligned pair), progressive alignment using a new profile function called the log expectation score, and refinement using tree-dependent restricted partitioning, has the clear and distinct advantage of speed. However, its accuracy is only intermediate as compared to ProbCons.

ProbCons takes an innovative approach to the sequence alignment problem by using an HMM. It does not incorporate any biologically relevant information at all (no position-specific gap scoring, no rigorous evolutionary tree construction, etc.). Due to its use of maximum expected accuracy as an objective function and the application of the probabilistic consistency transformation, it has the highest overall accuracy of the four programs reviewed. Its speed however, could be improved upon.

Finally, a word about COBALT. This program yielded speed and accuracy results that are about intermediate as compared to the four programs reviewed here. Because it is a constraint-based program, however, it offers something that many researchers may find exciting--the ability to input his/her own pairwise alignments.

An admittedly unscientific survey indicates that even with all of the new algorithms, many researchers still prefer CLUSTAL W (personal communication with members of the Stanford University Dept. of Biochemistry). This may be due, in part, to simple familiarity or an ignorance of the wealth of other, arguably better, MSA packages. The prudent researcher would do well to avail himself/herself of the many MSA programs at his/her disposal.

Figures

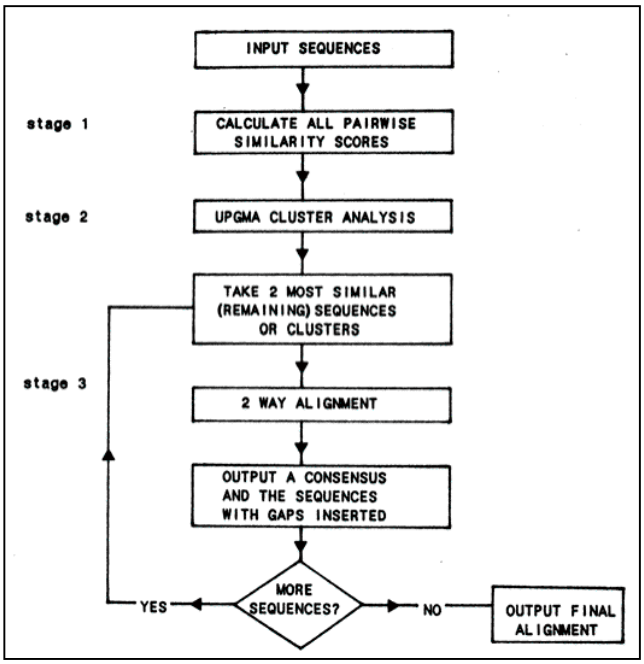

Figure 1. Flow chart of CLUSTAL MSA strategy as described in the text (Higgins and Sharp, 1988).

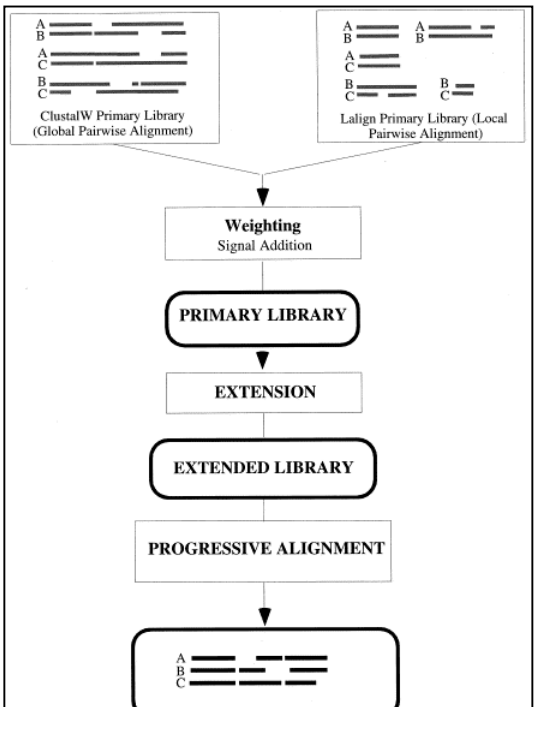

Figure 2. Layout of the T-Coffee strategy; the main steps required to compute a MSA using the T-Coffee method. Square blocks designate procedures while rounded blocks indicate data structures (Notredame et. al., 2000).

Figures, cont.

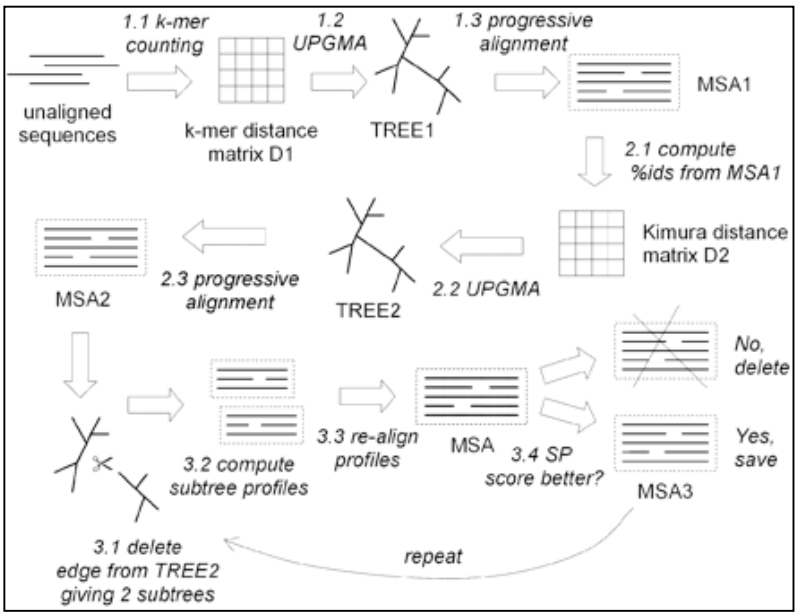

Figure 3. This diagram summarizes the flow of the MUSCLE algorithm. There are three main stages: Stage 1 (draft progressive), Stage 2 (improved progressive) and Stage 3 (refinement). A multiple alignment is available at the completion of each stage, at which point the algorithm may terminate (Edgar, 2004).

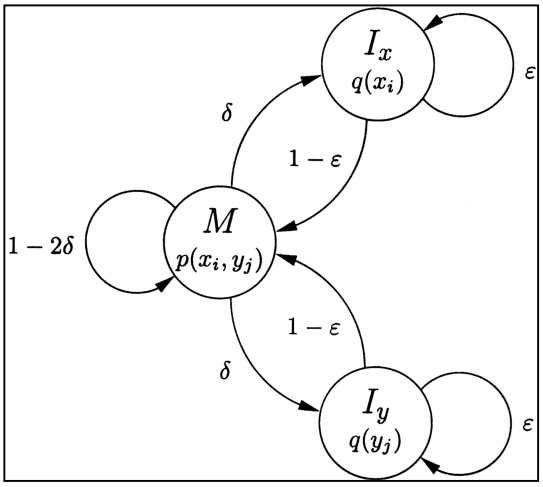

Figure 4. Basic pair-HMM for sequence alignment between two sequences, x and y. State M emits two letters, one from each sequence, and corresponds to the two letters being aligned together. State Ix emits a letter in sequence x that is aligned to a gap, and similarly state Iy emits a letter in sequence y that is aligned to a gap. Finding the most likely alignment according to this model by using the Viterbi algorithm corresponds to applying Needleman-Wunsch with appropriate parameters. The logarithm of the emission probability function p(.,.) at M corresponds to a substitution scoring matrix, while affine gap penalty parameters can be derived from the transition probabilities {delta} and {epsilon} (Do et. al., 2005).

Figures, cont.

Program	Publication date	Simprot	BAliBASE RV11	RV12	RV20	RV30	RV40	RV50	all refs.	Time (s)
Clustal W	Sep 1994	0.78923	0.48326	0.81114	0.82769	0.70009	0.65208	0.67851	0.70238	22.012
Dialign2.2	Mar 1999	0.75480	0.41433	0.77499	0.80789	0.66539	0.63704	0.66255	0.66723	52.708
T-Coffee	Sep 2000	0.83629	0.51866	0.84180	0.84176	0.73867	0.68560	0.74086	0.73186	1273.963
POA	Mar 2002	0.75196	0.30605	0.71622	0.77491	0.62705	0.58865	0.57238	0.60754	9.025
Mafft FFT-NS-2	Jul 2002	0.83911	0.46401	0.79774	0.83113	0.73048	0.64222	0.69748	0.70129	1
Muscle	Aug 2004	0.83031	0.53313	0.83181	0.84411	0.73635	0.66969	0.71056	0.73110	4.426
Mafft L-INS-i	Jan 2005	0.86545	0.56564	0.84497	**0.86049**	**0.77123**	**0.71307**	**0.75483**	0.75813	15.607
ProbCons	Feb 2005	**0.86712**	**0.59117**	**0.85479**	0.85796	0.76782	0.69439	0.75271	**0.76227**	353.787
Dialign-T	Mar 2005	0.77475	0.41372	0.79267	0.80824	0.67674	0.60237	0.67518	0.67024	41.467
Kalign	Dec 2005	0.80271	0.47593	0.82048	0.82854	0.72459	0.64190	0.70384	0.70801	3.403

Figure 5. Overall average accuracy values obtained with all Simprot's simulated sequences and all BAliBASE's references. Results are ordered by date of publication. Values in the same column that are not significantly different according to a Wilcoxon signed ranks test ($p < 0.05$) have the same color; values in black are significantly different, and bold font represents the largest value in the column. CPU times are normalized to Mafft FFT-NS-2 and were obtained with a 44-sequence alignment of 500 residues (Nuin et. al., 2006).

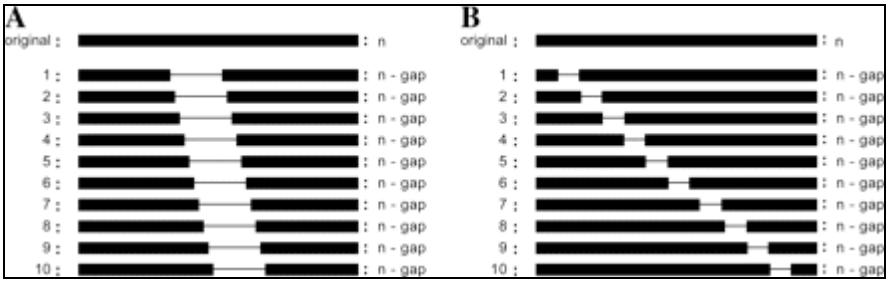

Figure 6. An amino acid sequence of size n was replicated 10 times, and a stretch of either 10 or 30 residues was deleted from each replicate, each time shifting the gap origin either (A) by one residue for overlapping deletions or (B) by the length of the deletion for non-overlapping deletions. A set of such gapped alignments was generated for each amino acid sequence, placing gaps along the length of the sequence (Golubchik et. al., 2007).

Figures, cont.

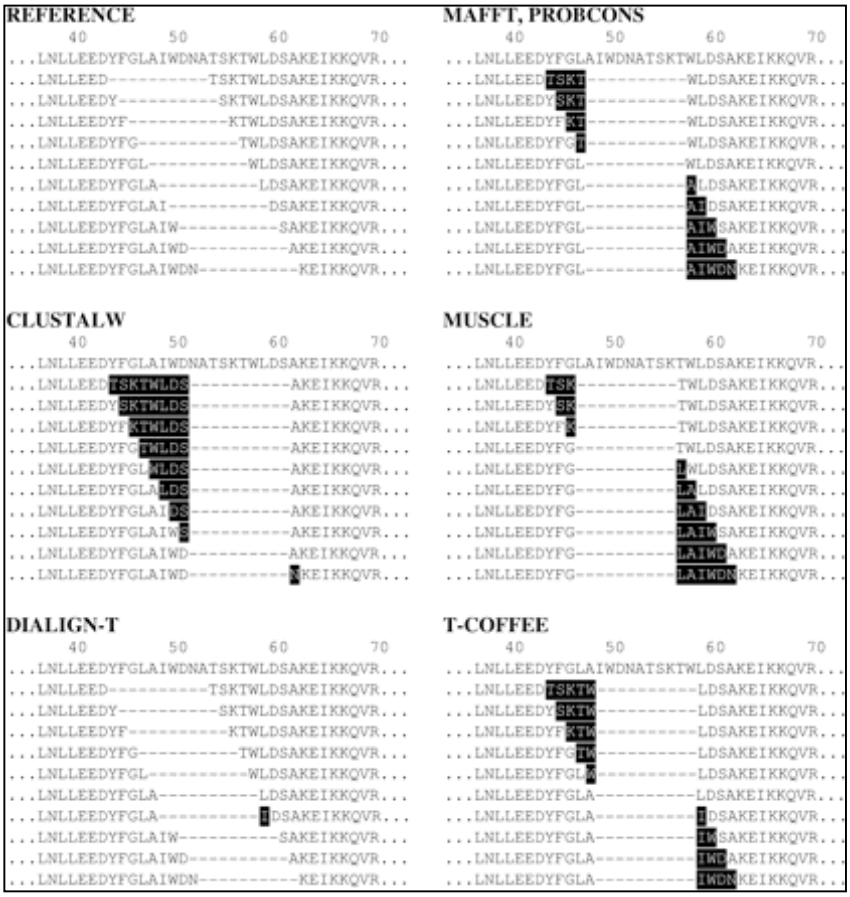

Figure 7. Summary view of alignments of highly similar sequences containing gaps (dashes) at overlapping positions. Misaligned residues are shaded. Alignments were generated by sequentially deleting a region of 10 residues from the EPB sequences and aligning the modified sequences with the original sequence using Clustal W, DIALIGN-T, MAFFT, MUSCLE, PROBCONS, and T-COFFEE. Results were compared with the reference alignment (REFERENCE) (Golubchik et. al., 2007).

Tables

Table 1. Performance of aligners on the BAliBASE benchmark alignments database

	Ref 1 (82)		Ref 2 (23)		Ref 3 (12)		Ref 4 (12)		Ref 5 (12)		Overall (141)		
Aligner	SP	CS	SP	CS	SP	CS	SP	CS	SP	CS	SP	CS	Time (mm:ss)
Align-m	76.6	n/a	88.4	n/a	68.4	n/a	91.1	n/a	91.7	n/a	80.4	n/a	19:25
DIALIGN	81.1	70.9	89.3	35.9	68.4	34.4	89.7	76.2	94.0	84.3	83.2	63.7	2:53
CLUSTALW	86.1	77.3	93.2	56.8	75.3	46.0	83.4	52.2	85.9	63.8	86.1	68.0	1:07
MAFFT	86.7	78.1	92.4	50.2	78.8	50.4	91.6	72.7	96.3	85.9	88.2	71.4	1:18
T-Coffee	86.6	77.4	93.4	56.1	78.5	48.7	91.8	73.0	95.8	90.3	88.3	72.2	21:31
MUSCLE	88.7	80.8	93.5	56.3	82.5	56.4	87.6	60.9	96.8	90.2	89.6	73.9	**1:05**
ProbCons	**90.1**	**82.6**	**94.4**	**61.3**	84.1	**61.3**	90.1	72.3	97.9	91.9	91.0	77.2	5:32
ProbCons-ext	90.0	82.5	94.2	59.1	**84.3**	61.1	**93.8**	**81.0**	**98.1**	**92.2**	**91.2**	**77.6**	8:02

Columns show the average sum-of-pairs (SP) and column scores (CS) achieved by each aligner for each of the five BAliBASE references. All scores have been multiplied by 100. The number of sequences in each reference is given in parentheses. Overall numbers for the entire database are reported in addition to the total running time of each aligner for all 141 alignments. The best results in each column are shown in bold (Do et. al., 2005).

Table 2. Performance of aligners on the PREFAB protein reference alignment benchmark

Aligner	Overall (1927)	Time
DIALIGN	57.2	12 h, 25 min
CLUSTALW	58.9	2 h, 57 min
T-Coffee	63.6	144 h, 51 min
MUSCLE	64.8	3 h, 11 min
MAFFT	64.8	**2 h, 36 min**
ProbCons	66.9	19 h, 41 min
ProbCons-ext	**68.0**	37 h, 46 min

Entries show the average *Q* (equivalent to SP) score achieved by each aligner on all 1927 alignments of the PREFAB database. All scores have been multiplied by 100. Running times for programs over the entire database are given for each program in hours and minutes. The best results in each column are shown in bold.

Tables, cont.

Table 3. Performance of aligners on the SABmark sequence and structure alignment benchmark

	Superfamily (462)		Twilight zone (236)		Overall (698)		
Aligner	f_D	f_M	f_D	f_M	f_D	f_M	Time (mm:ss)
Align-m	44.4	**58.9**	17.1	**43.0**	35.2	**53.5**	56:44
DIALIGN	50.3	42.5	22.5	19.2	41.0	34.6	8:28
CLUSTALW	53.7	38.7	24.8	15.2	43.9	30.8	**2:16**
MAFFT	54.1	40.0	24.8	16.0	44.2	31.9	7:33
T-Coffee	55.4	41.8	26.4	18.0	45.6	33.7	59:10
MUSCLE	55.9	40.1	27.6	17.5	46.4	33.0	20:42
ProbCons	**59.9**	45.0	**32.1**	21.7	**50.5**	37.1	17:20
ProbCons-ext	**59.9**	45.3	32.0	22.1	**50.5**	37.5	23:10

Columns show the average developer (f_D) score (equivalent to sum-of-pairs [SP] score) and modeler (f_M) score achieved by each aligner for the "Superfamily" and "Twilight Zone" sets in the SABmark database. All scores have been multiplied by 100. The number of sequences in each set is given in parentheses. Overall numbers for the entire database are reported in addition to the total running time of each aligner for all 698 alignments. The best results in each column are shown in bold (Do et. al., 2005).

Table 4**.** Root mean square deviation of Q-score for COBALT, Clustal W, MUSCLE, PCMA and ProbCons restricted to core regions on various benchmarks

Tool	Bali	HOM	IRM	PREFAB	SAB
COBALT versus ProbCons	6.71 (–s)	9.34 (s)	16.61 (0.8175)	16.39 (–s)	9.98 (0.1208)
COBALT versus PCMA	8.76 (–0.001)	11.60 (s)	15.27 (–s)	17.35 (–s)	12.99 (s)
COBALT versus MUSCLE	10.32 (0.0094)	10.07 (s)	51.66 (s)	16.48 (–0.0011)	13.05 (s)
COBALT versus Clustal W	16.88 (s)	12.37 (s)	76.92 (s)	20.89 (s)	14.51 (s)
ProbCons versus PCMA	7.02 (0.0011)	8.17 (s)	17.56 (–0.0017)	13.23 (s)	9.01 (s)
ProbCons versus MUSCLE	9.69 (s)	6.22 (s)	46.68 (s)	13.49 (s)	9.91 (s)
ProbCons versus Clustal W	17.31 (s)	9.75 (0.0004)	73.04 (s)	20.87 (s)	13.43 (s)
PCMA versus MUSCLE	10.97 (s)	7.94 (0.4581)	54.01 (s)	14.90 (0.0427)	9.46 (0.09)
PCMA versus Clustal W	17.32 (s)	5.93 (–0.1797)	79.14 (s)	18.97 (s)	11.77 (s)
MUSCLE versus Clustal W	13.21 (s)	8.64 (–0.1933)	36.73 (s)	18.36 (s)	10.41 (0.0059)

Significance is given in brackets and is calculated using Friedman's rank sum test, where the value '(s)' means a *P*-value of <1*E*–10. A negative *P*-value means that the method on the right performed better (had a lower average rank) than the method on the left (Papadopoulos and Agarwala, 2007).

References

Blackshields, G., Wallace, I. M., Larkin, M. and Higgins, D. G. (2006). Analysis and comparison of benchmarks for multiple sequence alignment. *In Silico Biology* **6**: 0300

Chenna, R., Sugawara, H., Koike, T. , Lopex, R. Gibson, T. J., Higgins, D. J. and Thompson, J. D. (2003). Multiple sequence alignment with the Clustal series of programs. *Nuclei Acids Research* **31**(13): 3497-3500.

Do, C. B., Mahabhashyam, M. S. P., Brudno, M. and Batzoglou, S. (2005). ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome Research* **15**: 330-340.

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

Edgar, R. C. and Batzoglou, S. (2006). Multiple sequence alignment. *Current Opinion in Structural Biology* **16**: 368-373.

Feng, D-F. and Doolittle, R.F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol* **25**: 351-360.

Golubchik, T. G., Wise, M. J., Easteal, S. and Jermiin, L. (2007). Mind the Gaps: Evidence of Bias in Estimates of Multiple Sequence Alignment. *Mo. Biol Evol* **24**(11): 2433-2442.

Higgins, D. G. (1988). CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* **73**: 237-244.

Hirosawa, M., Totoki, Y., Hoshida, M. and Ishikawa. M. (1995). Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput Appl Biosci* **11**: 13-18.

Morgenstern, B., Frech, K., Dress, A., & Werner, T. (1998). DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics* **14**(3), 290-294.

Notredame, C. (2002). Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics* **3**(1): 131-144.

Notredame, C. (2007). Recent Evolutions of Multiple Sequence Alignment Algorithms. *PLoS Computational Biology* **3**(8): 1405-1408.

Notredame, C., Higgins, D. G. and Heringa, J. (2000). T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *J Mol Biol* **302**: 205-217.

Nuin, P., Wang, Z. and Tillier, E. R. M. The accuracy of several multiple sequence alignment programs for proteins. *BMC Bioinformatics* **7**: 471-488.

Papadopolous, J. S. and Agarwala, R. (2007) COBALT: a constraint-based alignment tool for multiple protein sequences. *Bioinformatics* **23**(9): 1073-1079.

Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**(22): 4673-4680.

Thompson, J. D., Higgins, D. G., Plewniak, F. Jeanmugin, F. and Higgins, D. G. (1997). The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucleic Acid Research* **25**: 4876-4882.