MUSCLE Algorithm for Multiple Sequence Alignment

Nahian Salsabil
Student ID: 1705091

Muhtasim Noor
Student ID: 1705108

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

July 28, 2021

Contents

1 Introduction

This report briefly illustrates the well known *MUSCLE* Algorithm. The full form of MUSCLE is Multiple Sequence Comparison by Log-Expectation which is implemented for finding multiple sequence alignment(MSA).

Multiple alignments of protein sequences are important in many applications, including phylogenetic tree estimation, secondary structure prediction and critical residue identification. Two attributes of MSA programs are of primary importance to the user: biological accuracy and computational complexity (i.e., time and memory requirements). Complexity is of increasing relevance due to the rapid growth of sequence databases, which now contain enough representatives of larger protein families to exceed the capacity of most current programs. Obtaining biologically accurate alignments is also a challenge, as the best methods sometimes fail to align readily apparent conserved motifs.

MUSCLE is a new MSA program that provides significant improvements in both accuracy and speed.

2 Sequence Alignment

2.1 Sequence Alignment

In bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. The sequence alignment is made between one known sequence and one unknown sequence or between two unknown sequences. The known sequence is called reference sequence and the unknown sequence is called query sequence.

This is an example of sequence alignment.

2.2 Multiple Sequence Alignment

Multiple Sequence Alignment is generally the alignment of three or more biological sequence (Protein or Nucleic Acid) of similar length. This is a tool used to study closely related genes or proteins in order to find the evolutionary relationships between genes and to identify shared patterns among functionally or structurally related genes. MSA provides more information than pairwise alignments since they show conserved regions within a protein family which are of structural and functional importance.

Here is an example of MSA.

G	C	T	T	G	A	C	T	T	C	T	G	T
G	C	G	T	A	A	C	T	T	C	A	C	A
G	C	G	T	C	A	C	T	T	G	A	A	G
G	C	G	T	C	A	C	T	T	G	A	A	A
G	C	G	T	C	A	C	T	T	G	A	A	C

Figure 1: Multiple Sequence Alignment

In this image, all the aligned characters are colored accordingly.

3 Problem Solving Techniques

3.1 Primary Idea

The primary idea of solving multiple sequence alignment is *brute force.* But as it takes all possible alignments and checks whether the alignment is right or not, it is computationally expensive.

A common heuristic is to seek a multiple alignment that maximizes the SP score (the summed alignment score of each sequence pair)[3], which is NP complete. It can be achieved by dynamic programming with time and space complexity $O(L^N)$ in the sequence length L and number of sequences N, and is practical only for very small N. Stochastic methods such as Gibbs sampling can be used to search for a maximum objective score, but have not been widely adopted.

3.2 Improved Algorithm

In later years, a popular strategy named *progressive alignment* has been proposed which first estimates a phylogenetic tree. many MSA algorithms have been developed based on this strategy. The best-known of progressive algorithms is CLUSTALW[7], which is practical for up to a few hundred sequences on desktop computers.

One approach is to use a progressive alignment as the initial state of a stochastic search for a maximum objective score. Alternatively, pairs of profiles can be extracted from the progressive alignment and re-aligned, keeping the results only when an objective score is improved. MUSCLE is one of the algorithms which follows this approach and improves the speed significantly.

4 MUSCLE

4.1 Working Skeleton

MUSCLE works on the basis of progressive alignment[1] which is a procedure for generating an MSA that reduces the construction of the MSA to a series of pair-wise alignments. This process continues in an iterative fashion, adjusting the positioning of gaps in all sequences. The sequences are assumed a priori to share a common ancestor, and the trees are constructed from difference matrices derived directly from the multiple alignment. The thrust of the method involves putting more trust in the comparison of recently diverged sequences than in those evolved in the distant past. In particular, this rule is followed: "*once a gap, always a gap*".

4.2 Stages

MUSCLE algorithm basically works on two stages.

- Draft Progressive
- Improved Progressive

In the first stage, there will have some sequences of protein and amino acid and at the end of this stage those sequences will be aligned by executing progressive alignment resulting in a MSA.

In the second stage, that is improved progressive, the progressive alignment will be executed again on previously aligned sequences of stage one with different metrics. This stage attempts to improve the tree and builds a new progressive alignment according to this tree. At the completion of second stage, a multiple alignment is available and the algorithm can be terminated otherwise it may be iterated.

These two stages use different scoring metrics and algorithms to execute progressive alignment.

5 Progressive Alignment

Progressive alignment is a heuristic for multiple sequence alignment that does not optimize any obvious alignment score. The idea is to do a succession of pairwise alignments, starting with the most similar pairs of sequences and proceeding to less similar ones.

5.1 Similarity Measure

We calculate some score between every two sequences. The score for aligning a pair of sequences is determined by the profile function, which should assign a high score to pairs of sequences containing similar amino acids. The most similar sequences have the highest similarity score and the least similar sequences have the lowest.

5.1.1 Distance Matrix

Using the similarity scores, we calculate a matrix of distances between each pair of sequences. Consider this as an N-clique G, where edge {i, j} is labeled with the score of an optimal alignment of the i-th and j-th sequences.

5.2 Tree Construction

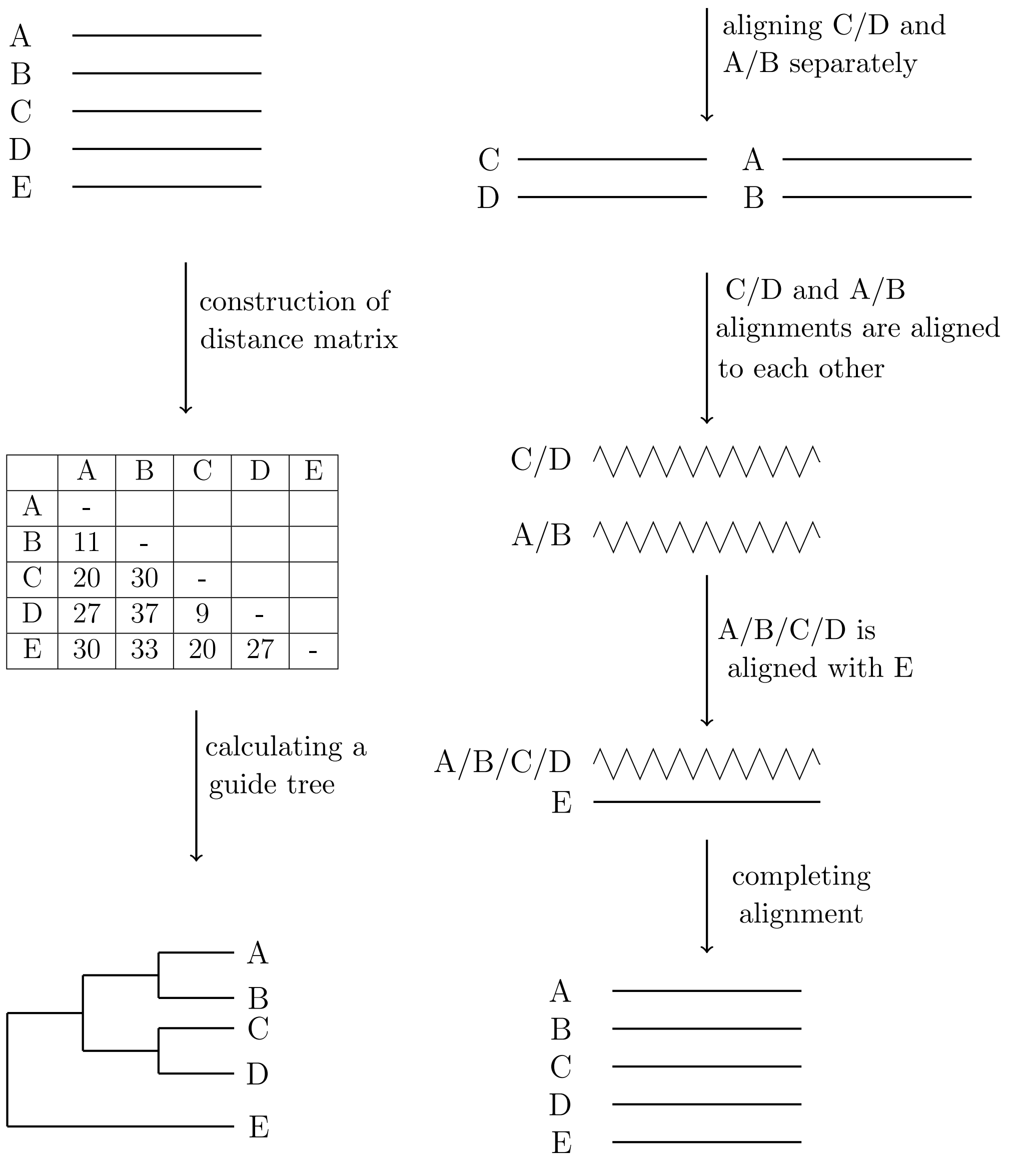
Then we use Kruskal's algorithm to find a minimum spanning tree of G. Whenever a minimum spanning tree edge would connect two components, instead we add a new root node with directed edges to the roots of the two components. This is the "guide tree". The method of building this guide tree is known as the "clustering method".

5.3 Alignment

A *Pairwise Alignment*[5] is used to identify regions of similarity that may indicate functional, structural and/or evolutionary relationships between two biological sequences (protein or nucleic acid).
We perform pairwise alignments according to the guide tree, working from the leaves to the root. A node u with children v and w corresponds to an alignment of the leaves of v's subtree (already aligned inductively) with the leaves of w's subtree (already aligned).

5.3.1 Multiple Sequence Alignment

As we keep on aligning sequences using the pairwise alignment method, we finally get an alignment of our initial sequences. This is the resultant multiple sequence alignment of our initial unaligned sequences.


A
B
C
D
E
construction of distance matrix
calculating a guide tree
A
B
C
D
E
aligning C/D and A/B separately
C
D
A
B
C/D and A/B alignments are aligned to each other
C/D
A/B
A/B/C/D is aligned with E
A/B/C/D
E
completing alignment
A
B
C
D
E

	A	B	C	D	E
A	-				
B	11	-			
C	20	30	-		
D	27	37	9	-	
E	30	33	20	27	-

Figure 2: Progressive Alignment (Step by Step)

6 Stage 1 : Draft Progressive

The first stage builds a progressive alignment. This initial alignment will be used in the second stage to build an improved alignment.

6.1 Similarity Measure

The similarity of each pair of unaligned sequences is computed, using k-mer counting.

A **κ-mer** is a contiguous subsequence of length k, also known as a word or k-tuple. Related sequences tend to have more k-mers in common than expected. The primary motivation for this measure is improved speed as no alignment is required.
We can compute the similarity between sequences X and Y as:

$$F = \sum_{\tau} min[n_X(\tau), n_Y(\tau)]/[min(L_X, L_Y) - \kappa + 1] \tag{1}$$

Here τ is a $\kappa - mer$, L_X, L_Y are the sequence lengths, and $n_X(\tau)$ and $n_Y(\tau)$ are the number of times τ occurs in X and Y respectively.

A triangular distance matrix is computed from the pairwise similarities.

6.2 Tree Construction

Given a distance matrix, a binary tree is constructed by clustering. MUSCLE uses **UPGMA**(unweighted pair group method with arithmetic mean)[2] method to build the tree.
MUSCLE implements three variants of UPGMA that differ in their assignment of distances to a new cluster.

- Average linkage :

$$d_{PC}^{Avg} = (d_{LC} + d_{RC})/2 \tag{2}$$

- Minimum linkage :

$$d_{PC}^{Min} = Min[d_{LC}, d_{RC}] \tag{3}$$

- Weighted mixture of minimum and average linkage:

$$d_{PC}^{Mix} = (1 - s)d_{PC}^{Min} + sd_{PC}^{Avg} \tag{4}$$

where s is a parameter set to 0.1 by default.

	A	B	C	D	E
A	-				
B	11	-			
C	20	30	-		
D	27	36	9	-	
E	30	33	20	27	-

Table 1: Distance Matrix

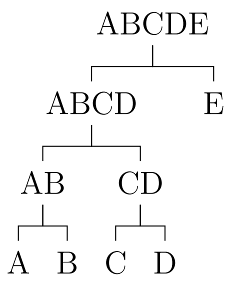


Figure 3: Tree Construction

6.3 Alignment

A progressive alignment is built by following the branching order of the tree. Sequences are assigned to the leaves of a binary tree. At each internal (i.e., non-leaf) node, the two child profiles are aligned using profile-profile alignment[6]. In a profile-profile alignment two profiles X and Y are aligned to each other such that columns from X and Y are preserved in the result. This procedure yields a multiple alignment of all input sequences at the root.

7 Stage 2 : Improved Progressive

The second stage attempts to improve the tree and builds a new progressive alignment according to the previously built tree. This stage may be iterated.

7.1 Similarity Measure

The similarity of each pair of sequences is computed using **Fractional Identity** computed from their mutual alignment in the current multiple alignment.
The Fractional Identity is computed as :

$$D = \frac{number\ of\ aligned\ characters\ in\ both\ sequences}{the\ total\ length\ of\ the\ sequences\ ignoring\ the\ gaps} \tag{5}$$

Sequence 1
A C T G A T C A T

Sequence 2
C C G C T C T A C

Alignment

1	2	3	4	5	6	7	8	9	10
A	C	T	G	A	T	C	-	A	T
C	C	-	G	C	T	C	T	A	C
	*		*		*	*		*	

$$Fractional\ Identity = \frac{5}{8}$$

7.2 Tree Construction

A tree is constructed by computing a **Kimura Distance Matrix**[4] and applying a clustering method to this matrix. The Kimura Distance is computed using the Fractional Identity of each pairs calculated in the previous step.
If D is the Fractional Identity, we use the following distance estimate:

$$d_{kimura} = -log_e(1 - D - D^2/5) \tag{6}$$

	Seq1	Seq2	Seq3	Seq4
Seq1	0	0.1	0.3	0.8
Seq2		0	0.2	0.7
Seq3			0	0.7
Seq4				0

$$\Big\downarrow \text{kimura}$$

	Seq1	Seq2	Seq3	Seq4
Seq1	0	0.1	0.38	0.26
Seq2		0	0.23	1.6
Seq3			0	1.6
Seq4				0

Table 2: Kimura Distance Matrix

7.3 Tree Comparison

Two trees are compared in order to identify those nodes that have the same branching orders within subtree rotation. If a progressive alignment has been created using to the old tree, then alignments at these nodes can be retained as the same result would be produced at those nodes by the new tree. New alignments are needed at the changed nodes only.
This procedure requires $O(N)$ time and space.

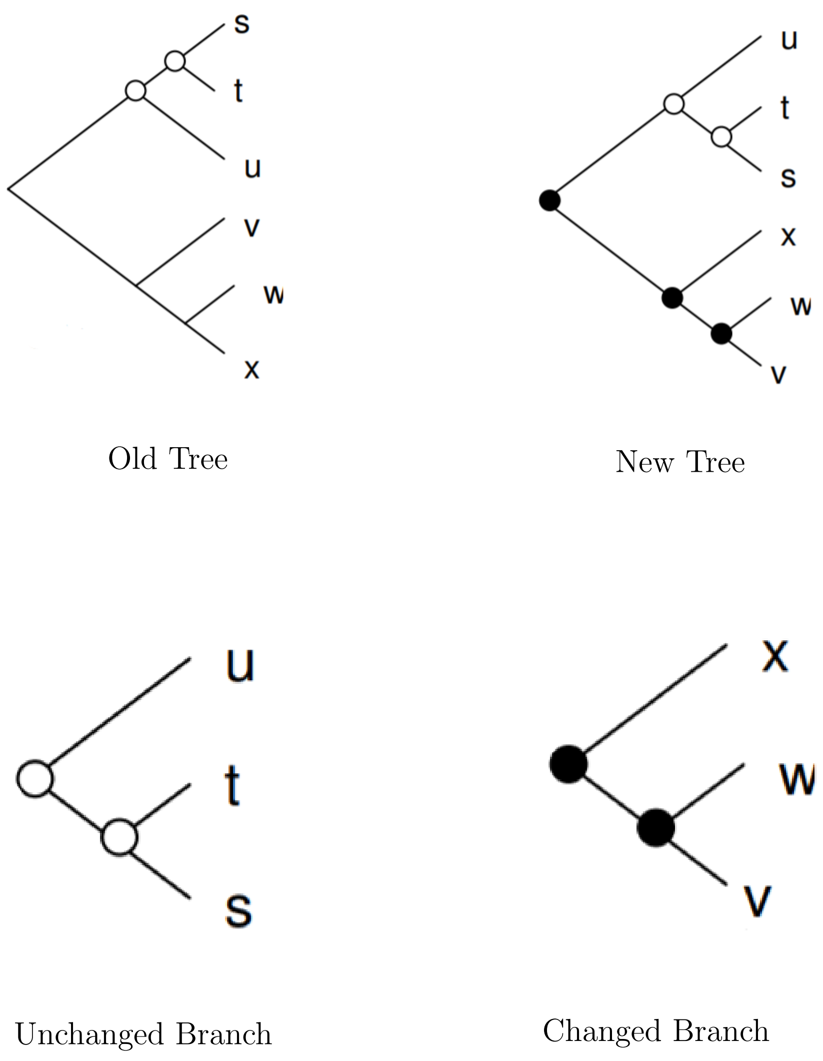


Figure 4: Tree Comparison

8 Termination

We repeat the whole process again and again until we get the unchanged root. On other words, The score of the multiple alignment which is calculated from distances implied by the new alignment is computed. If the score increases, the new alignment is retained, otherwise it is discarded. If all edges have been visited without a change being retained, or if a user-defined maximum number of iterations has been reached, the algorithm is terminated, otherwise it returns to step one of improved progressive. Visiting edges in order of decreasing distance from the root has the effect of first realigning individual sequences, then closely related groups.

9 Performance Analysis

- The memory complexity of MUSCLE is $O(Space) = N^2 + L^2$
- The time complexity of MUSCLE is $O(Time) = N^4 + NL^2$

Here is the performance graph.

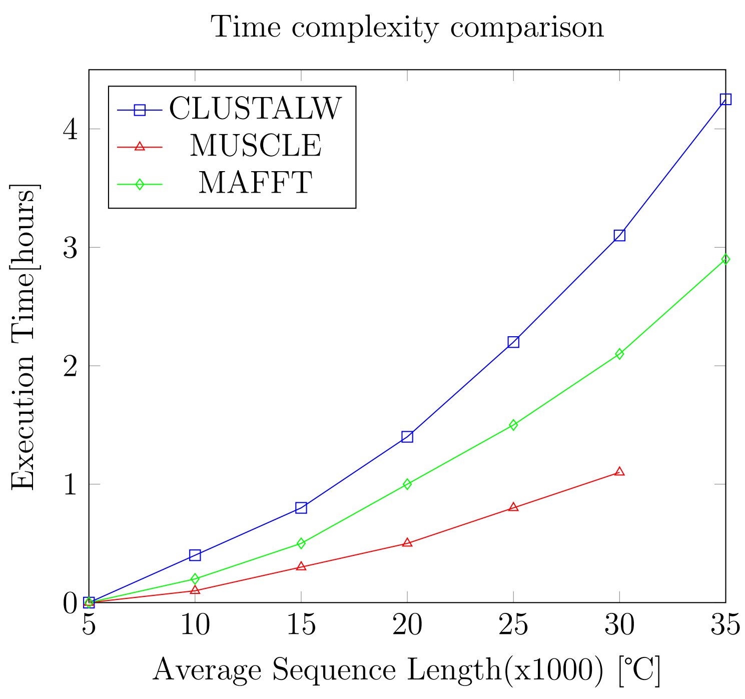


Figure 5: Performance Graph

This graph shows us that with the increasing length of sequences MUSCLE's execution time does not increase that much. So it can be said that MUSCLE is quite better than some of other algorithms.

10 Conclusion

MUSCLE demonstrates improvements in accuracy and reductions in computational complexity by exploiting a range of existing and new algorithmic techniques. While the design–typically for practical multiple sequence alignment tools–arguably lacks elegance and theoretical coherence, useful improvements were achieved through a number of factors. Most important of these were selection of heuristics, close attention to details of the implementation, and careful evaluation of the impact of different elements of the algorithm on speed and accuracy.

MUSCLE enables high-throughput applications to achieve average accuracy comparable to the most accurate tools previously available, which we expect to be increasingly important in view of the continuing rapid growth in sequence data.

MUSCLE is a command-line program written in a conservative subset of C++. At the time of writing, MUSCLE has been successfully ported to 32-bit Windows, 32-bit Intel architecture Linux, Solaris, Macintosh OSX and the 64-bit HP Alpha Tru64 platform. MUSCLE is donated to the public domain. Source code and executable files are freely available at http://www.drive5.com/muscle.

References

[1] Da-FeiFengRussell and F.Doolittle. “Progressive alignment and phylogenetic tree construction of protein sequences”. In: (2004). URL: https://doi.org/10.1016/0076-6879(90)83025-5.

[2] Ilan Gronau and Shlomo Moran. “Optimal implementations of UPGMA and other common clustering algorithms”. In: (2007).

[3] Winfried Just. “Computational Complexity of Multiple Sequence Alignment with SP-Score”. In: (2004). URL: https://doi.org/10.1089/106652701753307511.

[4] Motoo Kimura and Tomoko Ohta. “On the stochastic model for estimation of mutational distance between homologous proteins”. In: *Journal of Molecular Evolution* (1972). URL: https://link.springer.com/article/10.1007%2FBF01653945.

[5] Mehmet Koyutürk et al. “Pairwise Alignment of Protein Interaction Networks”. In: (2006). URL: https://doi.org/10.1089/cmb.2006.13.182.

[6] Niklas Von Öhsen, Ingolf Sommer, and Ralf Zimmer. “Profile-Profile Alignment : A Powerful Tool for Protein Structure Prediction”. In: (2002). URL: https://www.worldscientific.com/doi/abs/10.1142/9789812776303_0024.

[7] Julie D. Thompson, Toby. J. Gibson, and Des G. Higgins. “Multiple Sequence Alignment Using ClustalW and ClustalX”. In: (2002). URL: https://doi.org/10.1002/0471250953.bi0203s00.