Comparative Analysis of Multiple Alignment Algorithms for the Study of Multiple Sequence Alignment in Protein Nucleotides

Achennaki Shylla, Rajat Patel
Department of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
achen10@umbc.edu, rpatel12@umbc.edu
December 07, 2017

***Abstract*—: This paper explores comparative analysis of two most widely used algorithms for multiple sequence alignment of nucleotides like DNA, RNA and proteins. Our study compares these algorithms, their approaches i.e. progressive, iterative and progressive-iterative, and analyzes their outcomes for tackling multiple sequence alignment problems. We study these approaches through algorithms, Clustal Omega and MUSCLE. Further, performance of these multiple sequence alignment tools has been evaluated using standard gold benchmarks such as OXBENCH and PREFAB. The metrics that are used as objective function for evaluating these are Column Score, Structure similarity score, root mean square deviation, dependent average accuracy and quality score of the alignment. Practical time and space complexities for these algorithms have also been evaluated which could be informative to researchers in biological fields for better selection of algorithm based on the given type of input sequence which can range from simple protein amino acids to a human genome.**

***Keywords*— Multiple Sequence Alignment, Progressive approach, Iterative approach, Phylogenetic analysis, sequence alignment tools, Clustal Omega, MUSCLE, time and space complexity.**

I. Introduction

The field of bioinformatics deals with the study of automatic information processing of biological macromolecules [3]. This information process is crucial for phylogenetic analysis inclusive of tasks like function and structure prediction using macromolecules such as DNA/RNA and proteins. Comparison of biological sequences plays a major role in processing this information and helps in the study of evolutionary relationships that exist between species and to determine the homologous nature of two or more macromolecules, for example: the existence of a common biological ancestor. The comparison of biological sequences can be achieved by sequence alignment which involves matching of substrings within these sequences keeping same order of occurrence [3]. Sequence alignment can be performed manually for shorter and homogeneous sequences. Any larger sequence would require computational methods based on the length of sequence and the complexity of alignment. These include global and local methods. Global alignment spans the entire sequence and deal with similar pair of sequences while local alignment spans regions of similarity between dissimilar pair of sequences. Hybrid cases of such alignments exist in the form of semiglobal alignment. When aligning more than one query sequence, pairwise and multi alignment methods come into picture.

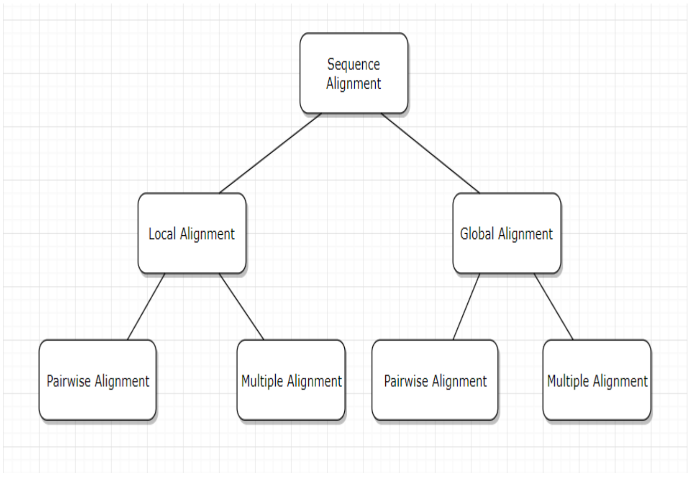


Figure 1: Sequence Alignment methods.

Pairwise alignment methods check for a matching local or global alignment of two query sequences in parts while multiple alignment methods handle two or more sequence queries. Further, pairwise global alignments algorithms employ two kinds of approaches, namely, dynamic programming approach and the anchoring approach while pairwise local alignment algorithms employ dynamic programming and the seeding approach [3]. Pairwise sequence alignment methods are efficient for similar sequences. Multiple sequence alignment methods prove useful when dealing with two or more query sequences, thus extending pairwise sequencing. Multiple sequence alignments are known to be difficult and may produce NP-complete problems due to genetic mutation in the genomes of living organisms. In this paper, we will be dealing with multiple sequence alignment algorithms and their approaches. Although there are multiple approaches, we will be evaluating two prominent approaches progressive and progressive-iterative.

II. Background

Multiple sequence alignment refers to the process of aligning three or more biological sequences. The four basic approaches of aligning multiple sequences are Dynamic Programming, Progressive approach, Iterative approach and Divide and Conquer approach. This section provides a brief overview of these approaches.

a. *Dynamic Programming approach:* Dynamic programming technique for multiple sequence alignment involves sets of two parameters, a gap penalty and substitution matrix. This approach uses a simple substitution matrix for capturing the similarity measure between the nucleotides sequences. However the search space for aligning sequence based on the constructed matrix increases exponentially. The runtime of this approach is strongly dependent on the length of the sequences. The time complexity of dynamic programming approach is given as $O(L^N)$, where L is the length of sequences for multiple sequences alignment and N is the number of sequences used for alignment [3].

b. Progressive approach: Progressive approach is one of the most popular approach for multiple sequence alignment. Through this technique the alignment of sequences takes place in gradual manner. A set of N sequences is taken as input as an initial step pairwise alignment is performed. A score function calculated between the pair of sequences using score function such as sum of pairs, distance between the pair is calculated to form a substitution matrix. Finally, the final multiple alignment is performed by using the algorithm based on pairwise alignment obtained. The two most similar sequences are aligned at each iteration which is selected based on the score function. Thus, we align larger and larger sequences gradually until all N sequences are aligned [3].

c. *Iterative approach*: Iterative approach as the name suggest constructs an initial alignment then at each iteration certain kinds of modification on the current alignment are performed based on the profile function to optimize the quality of the alignment and to get a better accuracy. Thus, the technique converges once there are no further improvement in the current alignment. [3]

d. *Divide and Conquer approach*: The Divide and Conquer approach algorithms processes the sequences to align simultaneously. There are three stages of operation in this technique [3]. In the first stage the position for dividing each sequence is selected, thus dividing the sequences into smaller sequences: a prefix and suffix as two parts. In the second stage, the sequences are divided recursively until we obtain small sequences that can be aligned optimally. In the final stage the sequence alignment is obtained by concatenating the alignments of the smaller sequences.

III. Methodology

This section describes the two multiple sequence alignment algorithms studied in this paper Clustal Omega and MUSCLE algorithm. Both these algorithms are most widely used today in the field of computational biology for study of nucleotides and their phylum structures.

1. *Clustal Omega or Clustal* Ω

Clustal Omega is the latest version in the Clustal series of Multiple Sequence Alignment tools. Clustal Omega is built to procure accurate alignment results for over 10,000 sequences of DNA, RNA or protein. In Clustal Omega, guide trees are calculated using mBED algorithm, enabling it to handle large datasets of biological sequencing. In addition, alignment of hidden Markov Models via the HHalign method helps to improve accuracy [4]. These are the major improvements made from the previous ClustalW version. The optimal alignment between N sequences generates a complexity of $O(L^N)$. The length for N sequences is given by L. Progressive Alignment methods usually yield a complexity of approximately $O(N^2)$ [1]. This approach employs a greedy strategy, hence, errors in the alignment initially calculated are irreversible. Clustal Omega is a tool which not only operate on large alignments but is also focused towards accuracy of alignment. The complexity of Clustal Omega is $O(NLogN)$ obtained by implementing modified mBED version [1].

Clustal Omega constructs a guide tree for unaligned sequences and integrates consistency for the same. Progressive alignment is executed on the guide tree simultaneously informing mergers of the alignment of the same [4]. This is the default mode of Clustal Omega. For very large sequences iterative refinement can be performed if desired. Hidden Markov Model profiles are constructed in such cases and alignment is re-executed. Another prominent feature is the External Profile Alignment (EPA) which predicts the final alignment and creates a hidden Markov Model for the same [1]. This helps to overcome the drawback observed from pairwise alignment of progressive approach which is not executed in parallel, hence the drag in speed of execution. Clustal Omega employs two hidden Markov Model alignment algorithms i.e. the Maximum Accuracy (MAC) algorithm and the Viterbi algorithm [4]. The default algorithm is MAC which has a high rate of memory consumption proven for accuracy. The Viterbi algorithm runs once exhaustion of memory resources are detected.

The mBED [6] algorithm involves three steps:
(a) *Select seeds as references*:
Consider a dataset D, with s sequences as sample. Then by Linial-London-Rabinovich (LLR) algorithm [11]:

$$s = (\log_2 N)^2$$

Let X be the reference point, then compare seeds and remove shorter ones if they are identical.

(b) *Potential seed selection*:
Use the reference X for embedding input sequences. Progressive approach can be used to find extra seeds to characterize D better. There are two approaches for this: '*use Pivot Objects*' heuristic – finds sequence outliers using seeds

and identifies the sequence furthest as new seed. '*use Pivot Groups*' heuristic – finds groups instead of sequences.

(c) *Embed inputs*:
Post process of choosing the seeds in X, associate all sequences in D with an s-dimensional vector by calculating the distance between s seed and the sequences. The embedded vector takes this distance as coordinate values.

mBED algorithm

```
/'usePivotObjects' heuristic/
1 $s = (\log_2 N)^2$
2 for seed x in X:
3     Let y be the sequence in D that maximizes $d(y, x)$.
4     Let z be the sequence in D that maximizes $d(z, y)$.
5     Return z as a new seed
6 for sequence $x$ in $D$
7     $F(x) = [d(x, X_1), d(x, X_2) \dots d(x, X_s)]$.
```

Or

```
/'usePivotGroups' heuristic/
1 $s = (\log_2 N)^2$
2 for seed x in X:
3     Let y be the sequence in D that maximizes $d(y, x)$.
4     Let z be the sequence in D that maximizes $d(z, x) +$
      $d(z, y)$.
5     Let w be the sequence in D that maximizes
      $d(w, x) + d(w, y) + d(w, z) \dots etc.$
6 for sequence $x$ in $D$
7     $F(x) = [d(x, X_1), d(x, X_2) \dots d(x, X_s)]$.
```

Pseudo-Code for mBED algorithm [6]

2. *MUSCLE Algorithm:*
The MUSCLE Algorithm is implemented basically in three stages: (a) Draft progressive alignment, (b) Improved progressive alignment, (c) Refinement stage.

a. *Draft progressive alignment:*
In this stage, a progressive alignment is formed using a similarity measure which is computed using k-mer distance [5]. The similarity measure from kmer is a contiguous subsequence of length k, which can also be denoted as k-tuple. Related sequences tend to have more k-mers in common while the divergent sequences k calculated is not that great [13]. The primary motivation of similarity measure is to improve the measure speed as no alignment is required. The similarity measure between the sequences X and Y can be given as:

$$F = \Sigma\tau min\ [nX(\tau), nY(\tau)] \ / \ [min(Lx, Ly) - k+] \quad (1)$$

Here τ is a k-mer, Lx Ly are the sequence lengths, and nX (τ) and nY(τ) are the number of times τ occurs in X and Y respectively. The denominator of F is given by the maximum number of k-mers that could be aligned. Thus, now given the similarity value we can measure additive distance between any two sequence (A, B) is the sum of the distance measure of combination of three sequences (A, B, C).

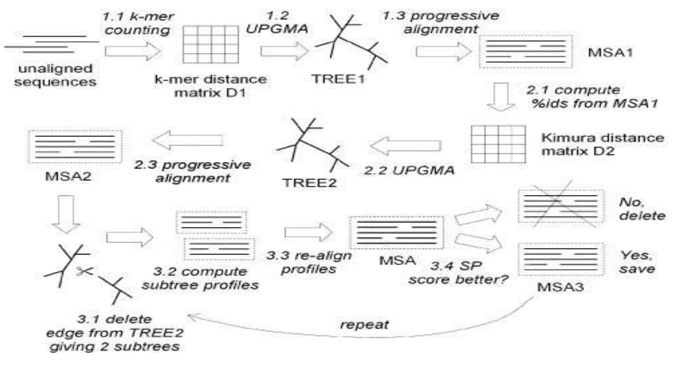
$$D(A, B) = d(A, C) + d(B, C) \quad (2)$$

The distance calculated here is the mutation distance (D) between sequence pairs that describes the historical path through the phylogenetic tree. The kmer distance is computed for each pair of input which would give the distance matrix D1. Using a UPGMA clustering algorithm, matrix D1 is clustered to produce binary tree TREE1. Thus, during this phase a root is identified for this particular tree formed. After a progressive alignment of the sequences is performed by branching the order of TREE1. A profile is constructed using a profile function at each leaf node of the binary tree TREE1. Thus, while traversing the tree is traversed in prefix order. At each child node, a pairwise alignment is constructed for two child profiles. Thus, alignment is then aligned with the parent node, therefore all nodes finally align at the root node to give a multiple sequence alignment.

The profile function calculated as given in [7] is computed as Log expectation measure which is a modified form of Log average measure as given in [12]. The profile function is given by the following equation.

$$LE^{xy} = (1 - f_G^x)(1 - f_G^y) \log \textstyle\sum_i \sum_j f_i^x f_j^y p_{ij} / p_i p_j \quad (3)$$

The equation gives Log expectation of i and j amino acid types, p_I is the background probability of i amino acid and p_j is the background probability of j amino acid and p_{ij} is the joint probability of i and j being aligned to each other. The values of f_G^x and f_G^y are the frequency of i and j occurring in columns x and column y respectively.



1.1 k-mer counting
1.2 UPGMA
1.3 progressive alignment
unaligned sequences
k-mer distance matrix D1
TREE1
MSA1
2.1 compute %ids from MSA1
Kimura distance matrix D2
2.2 UPGMA
TREE2
2.3 progressive alignment
MSA2
No, delete
Yes, save
3.3 re-align profiles
MSA
3.4 SP score better?
MSA3
3.2 compute subtree profiles
3.1 delete edge from TREE2 giving 2 subtrees
repeat

Figure 2: An overview of the working of the MUSCLE algorithm [7]

b. *Improved Progressive.*
In Improved progressive alignment a more accurate distance measure is used to find the similarity measure between the alignments called as kimura distance. The calculation of kimura distance is also performed to better the alignment formation

which is formed in stage 1 using k-mer distance which may be prone to inaccuracy since the distance calculation is approximate. Thus, now Kimura distance is calculated as given in [14] to give a matrix D2.

The Kimura distance is calculated as follow as

$$d_{kimura} = -\log_e\left(1 - D - \frac{D^2}{5}\right) \quad (4)$$

As in stage 1 the matrix D2 is clustered using UPGMA to produce a binary tree TREE2. Thus, a similar traversing of tree is performed as performed in stage 1 to produce a progressive alignment. Further, the progressive alignment formed is optimized by computing the alignment of the subtrees whose branching order changes with reference to TREE1.

c. *Refinement stage*

In the refinement stage each edge is traverse in TREE2 in breadth first search format, to divide into subtrees. A profile of multiple alignment is computed for each subtree. As in stage one the sequences are re-aligned to form a new multiple alignment and profile function of the new alignment is calculated if the profile function score is improved then the alignment is kept otherwise the alignment is deleted. This iteration is computed until convergence or until user define iterations. [7].

The theoretical time complexity for MUSCLE algorithm is $O(N^4 + L^2)$ and the theoretical space complexity is given as$O(N^2 + L^2)$[7].

IV. Implementation

To perform multiple sequence alignment on larger amount using Clustal Omega and MUSCLE algorithm we provisioned a RHEL Linux 7.0 server on Google Cloud Platform for better performance and easy accessibility for this project. Clustal Omega and MUSCLE tool were downloaded onto the server. The benchmark suits which are databases with various types of proteins nucleotides were installed on this server along with their libraries for data extraction.

The implementation of multiple sequence alignment algorithm for comparative analysis is done initially by taking a protein sequence from BLAST database where the protein sequence was identified to carry out sequence alignment. For this purpose, a sequence of amino acid sequence was identified along with various variations of the sequence. The amino acid sequence has around 1000 nucleotides and 10 such sequences were taken for implementation. These sequences were downloaded in FASTA format from the BLAST database. Clustal Omega and MUSCLE applications were used from the command line for aligning the sequences. The sequences aligned have the objective function of sum of pairs and column score. A tool called Unigene [10] has been used for viewing the aligned sequences and the dendrogram of the clusters that are formed in the MUSCLE and Clustal Omega algorithms.

Below are the alignment output of the sequences and the Dendrogram represent the tree that is formed using each algorithm:

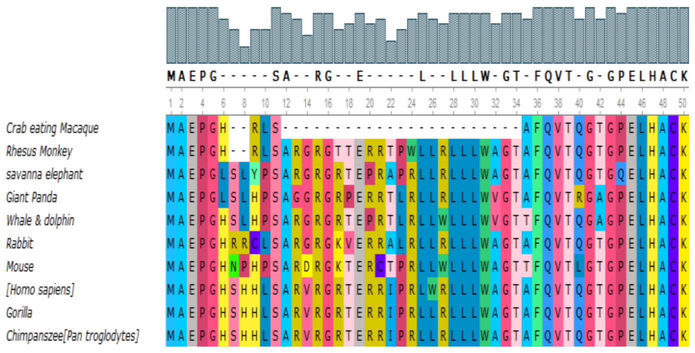


Figure 3: Multiple alignment using Clustal omega for 10 sequences (first 50 nucleotides) on UNIGENE

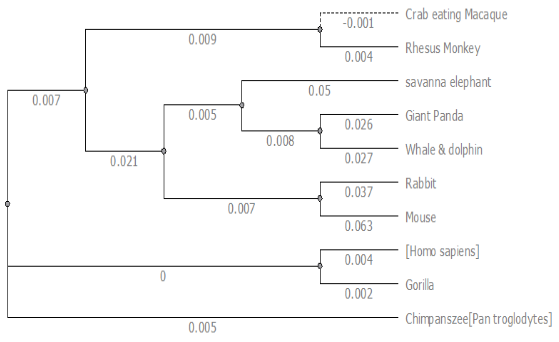


Figure 4: Dendrogram of Clustal Omega formed for multiple sequence alignment on UNIGENE.

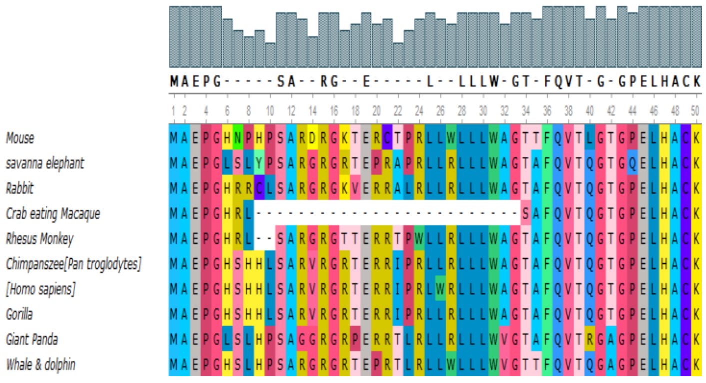


Figure 5: Multiple alignment using MUSCLE for 10 sequences (first 50 nucleotides) on UNIGENE

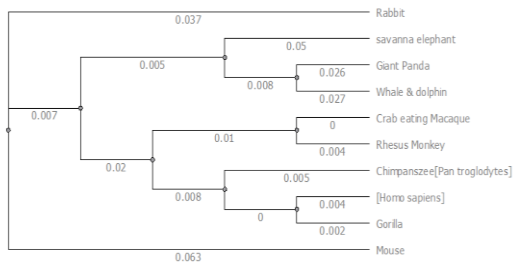


Figure 6: Dendrogram of MUSCLE formed for multiple sequence alignment on UNIGENE.

Thus, after a test implementation of the algorithm we selected sequences from OXBENCH benchmark suit. Using the unaligned sequences in the OXBENCH suit, we selected out top 45 unaligned sequences which had average length greater than 100 for each sequences and number of sequences ranging from 5 to 299 in a protein.

The time and space complexity for aligning these sequences were captured at OS level for both algorithm during the alignment process. The time complexity versus the length of sequences data was analyzed to give output as shown in the following figures. Time complexity measure against sequence length was relative as many sequences had smaller length but the number of sequences was high. Time versus space was also analyzed for both the algorithms.

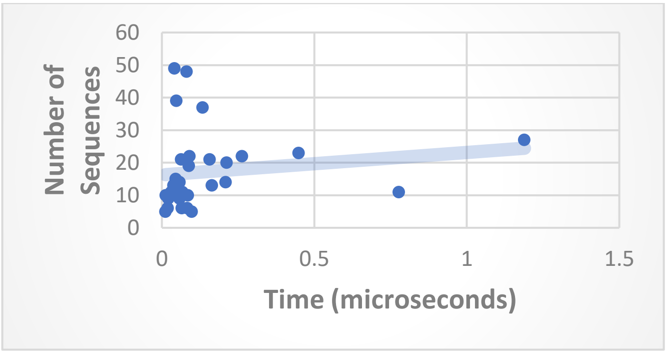


Figure 7: CLUSTAL OMEGA: Time vs number of sequences.

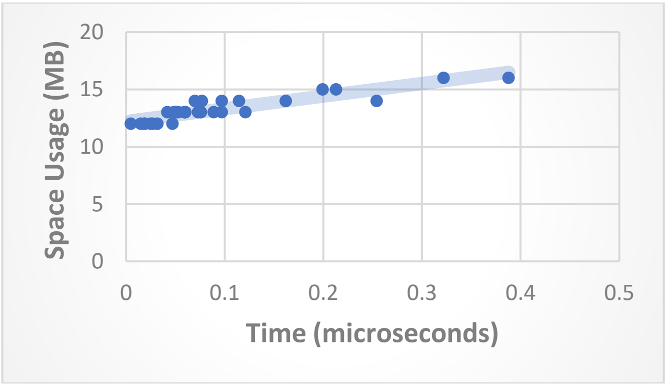


Figure 8: CLUSTAL OMEGA: Space usage.

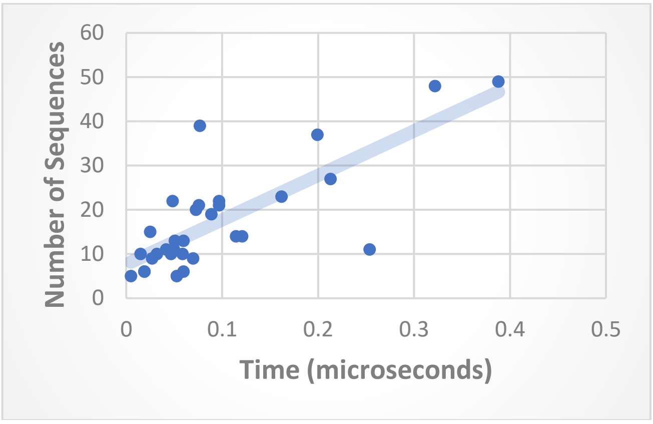


Figure 9: MUSCLE: Time vs number of sequences.

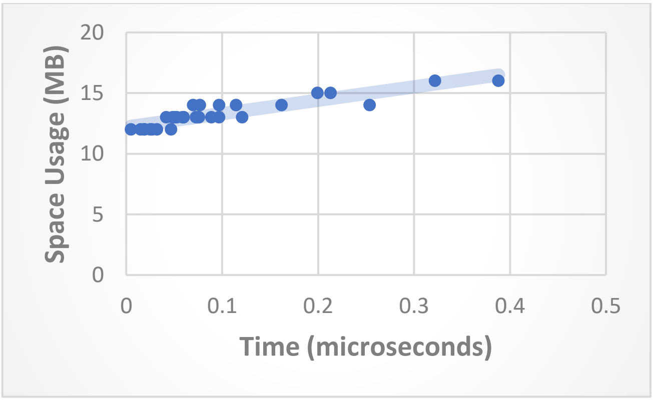


Figure 10: MUSCLE: Space usage.

V. Benchmark Analysis

Benchmark Analysis for algorithms Clustal Omega and MUSCLE algorithm was performed using OXBENCH and PREFAB benchmark suits. OXBENCH suite is a database with reference alignments and software tools for assessing the progress in the sequence alignment. Using OXBENCH data suite, following benchmark metrics were calculated for the same set of sequences on which alignment was performed [9]

1. *Structural Alignments*(S_c)

Using the multiple structure comparison algorithm applicability in OXBENCH the structure similarity between the test and reference alignments was calculated. The structure similarity score was computed through STAMP algorithm as given in [15]. The algorithm provides a threshold value of 3.0 for structural similarity score (S_c) above which alignment indicate clear structure similarity.

For calculating structure similarity, initially average sum of pair is calculated using the following equation:

$$S_p = \sum_{i=1}^{n} \frac{P_{ij}}{n} \tag{5}$$

The value of P_{ij} is the probability of two structure *i* and *j* being similar number of equivalent pairs.
Thus, using following equation we can calculate the value of structural similarity.

$$S_c = \left(\frac{S_P}{n}\right)\left(\frac{n-I_a}{L_a}\right)\left(\frac{n-I_b}{L_b}\right) \qquad (6)$$

S_c = Structural Similarity Score
n = number of equivalent residue, L_a= length of alignment a, L_b= length of alignment b, I_a= length of gap introduced in a, I_b= length of gap introduced in b.

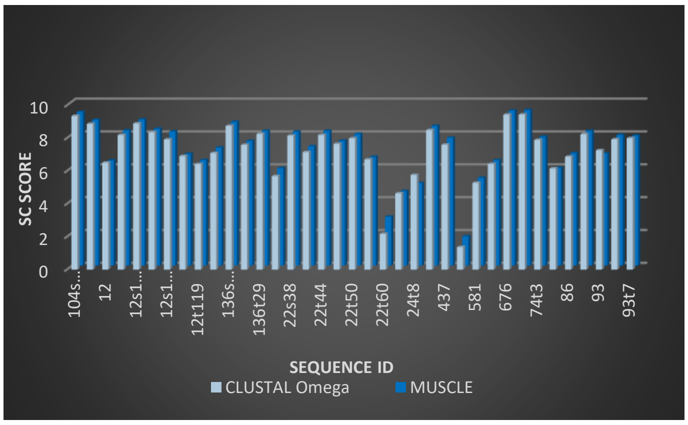


Figure 11: Structural similarity comparison for aligned sequence using Clustal omega and MUSCLE algorithm

2. *Root Mean Square Deviation:*

Root mean square deviation (rmsd) is standard deviation calculated for multiple sequence alignment. This measure is independent of the reference sequences. This measure provides the error deviation of the sequence alignment after aligning with Clustal Omega and MUSCLE algorithm. Comparative results show that error deviation in sequences aligned with Clustal Omega is more than the MUSCLE algorithm,

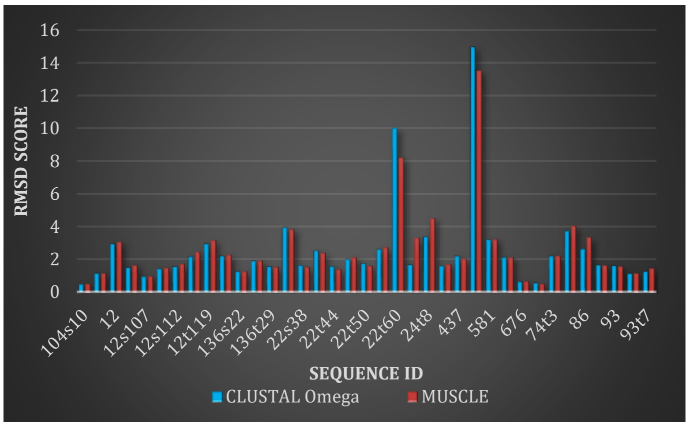


Figure 12: Comparing Root mean square deviation calculated for all the sequence using Clustal Omega and MUSCLE Algorithms.

3. *Dependent Average Accuracy(dep):*

The average dependent accuracy measure is computed by comparing the reference and aligned sequences with Clustal Omega and MUSCLE algorithm. The accuracy for multiple sequence alignment is calculated with following equation [9]

$$DEP = 100 * \left(\frac{Number\ of\ correctly\ aligned\ sequence}{Length\ of\ reference\ alignment}\right) \qquad (7)$$

We analyzed the accuracy of the aligned sequences by plotting the score of accuracy for aligned sequences obtained from both the algorithms on the same graph. We examined that accuracy was higher for aligned sequences using MUSCLE than Clustal Omega.

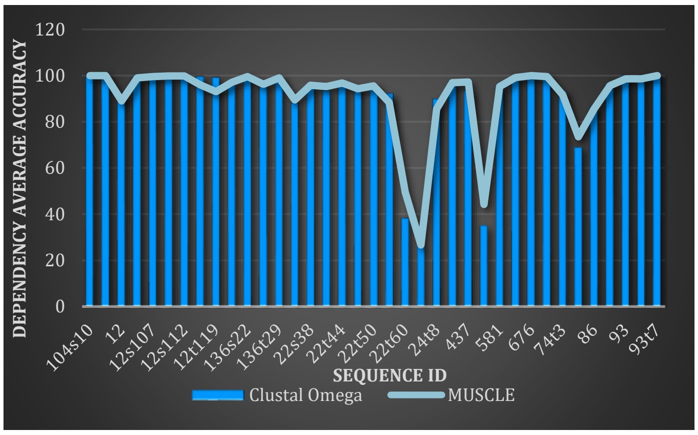


Figure 13: Comparison of Dependent average accuracy (dep) for Clustal Omega and MUSCLE aligned sequences

We also examined the relation between accuracy versus the average sequence length, however comparable readings were observed between the two algorithms results as they both performed fairly the same for a particular length of sequences.

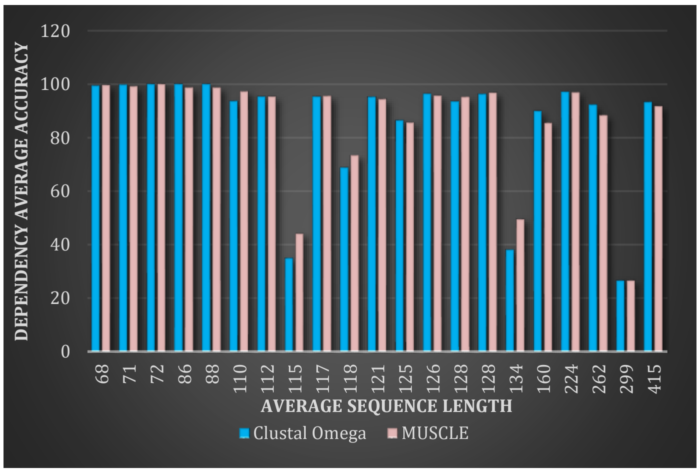


Figure 14: Analysis of relation between average accuracy and average length of sequence for alignment.

Column score: Accuracy of columns is calculated between the reference alignment and the test alignment. The percentage of identical columns is then calculated [9].

$$Column\ score = \frac{\text{Number of correctly aligned columns}}{\text{Total number of columns}} \quad (8)$$

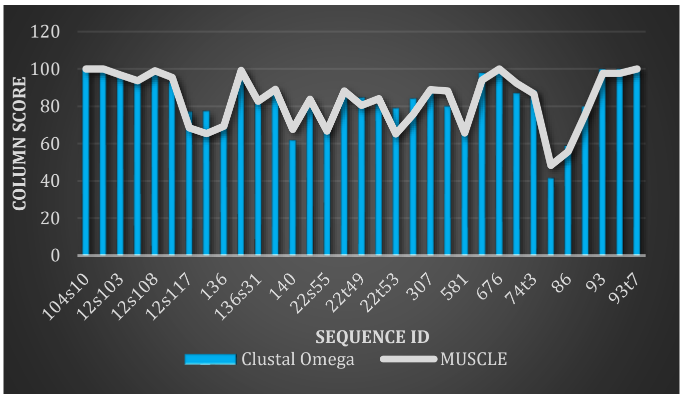


Figure 15: OXBENCH Column Score Accuracy: CLUSTAL Omega vs MUSCLE

PREFAB: Evaluates alignments based on quality (Q) score [7]. Q score metric calculated as follows:

$$\text{Q score} = \frac{\text{Number of correctly aligned residue pairs}}{\text{Total number of residue pairs}} \quad (9)$$

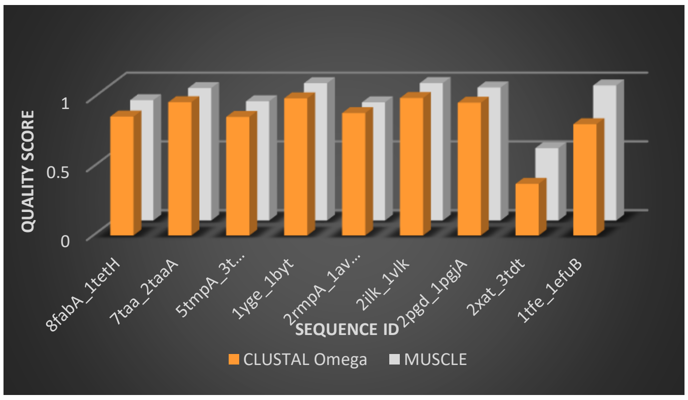


Figure 16: Prefab Quality Score: CLUSTAL Omega vs MUSCLE

VI. Conclusion

Based on observations of performance measure against benchmark metrics, MUSCLE edges out CLUSTAL Omega in terms of speed and accuracy. While space usage is comparable, in terms of speed, although comparable for shorter sequence length, MUSCLE is faster for larger sequences. Based on the dependent average accuracy metric, we observed that MUSCLE provides better percentage for accuracy in alignment. Further, although the structure similarity scores are comparable, the root mean square deviation scores are higher in CLUSTAL Omega readings which suggests that error rates while determining accuracy in alignment are higher for CLUSTAL Omega, hence MUSCLE edges out in this part as well. The column score further boosts the rating for MUSCLE in terms of accuracy in alignment as observed from the benchmark readings. The benchmark readings for Quality score on PREFAB further justifies that MUSCLE outperforms CLUSTAL Omega and is the preferred multiple sequence alignment tool.

ACKNOWLEDGEMENT

The authors would like to thank Dr.Alan T. Sherman for his guidance and insightful comments towards the project, more so, towards conveying algorithmic analysis techniques. Special mention to Dr. Ivan Erill, Department of Biology, UMBC for providing insight towards biological sequencing and Dr. Geoffrey J Barton, School of Life Sciences, University of Dundee for their contribution towards benchmark analysis. Also, acknowledging our reviewers Khurana and Saha for their valuable feedback.

References

[1] Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins DG. 2011. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol 7:539. 10.1038/msb.2011.75 http://msb.embopress.org/content/msb/7/1/539.full.pdf

[2] Pais, F. S.-M., Ruy, P. de C., Oliveira, G., & Coimbra, R. S. (2014). Assessing the efficiency of multiple sequence alignment programs. Algorithms for Molecular Biology : AMB, 9, 4. http://doi.org/10.1186/1748-7188-9-4

[3] Elloumi, Mourad, and Albert Y. Zomaya. Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications. Wiley, 2011.

[4] Rusell, J. David. Multiple Sequence Alignment Methods. Humana Press, 2013.

[5] Edgar RC. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004;5:113. doi:10.1186/1471-2105-5-113.

[6] Blackshields G, Sievers F, Shi W, Wilm A, Higgins DG (2010) Sequence emBedding for fast construction of guide trees for multiple sequence alignment. Algorithms Mol Biol 5: 21

[7] Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic

[8] Acids Research. 2004;32(5):1792-1797. doi:10.1093/nar/gkh340.

[9] Raghava GP, et al. OXBench: a benchmark for evaluation of protein multiple sequence alignment accuracy, BMC Bioinformatics , 2003, vol. 4 pg. 47

[10] Pontius, J.U., Wagner, L., and Schuler, G.D. 2002. Part 3. Querying and linking the data, 21. UniGene: A unified view of the transcriptome. In the NCBI Handbook (internet; ed. J. McEntyre), National Library of Medicine, Bethesda, MD.

[11] Linial N, London E, Rabinovich Y: The Geometry of Graphs and Some of its Algorithmic Applications**.** *Combinatorica* 1995, 15:215-245

[12] Sjolander,K., Karplus,K., Brown,M., Hughey,R., Krogh,A., Mian,I.S.and Haussler,D. (1996) Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology. CABIOS,12, 327±345.

[13] Edgar,R.C. (2004) Local homology recognition and distance measures in linear time using compressed amino acid alphabets. Nucleic Acids Res.,32, 380±385.

[14] Kimura,M. (1983) The Neutral Theory of Molecular Evolution. Cambridge University Press.

[15] Russell RB and Barton GJ: Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. Proteins 1992, 14:309-323.