Optimal Algorithm for DNA Multiple Sequence Alignment

Jinal Mashruwala	Krishit Shah	Poornima Kumar
(UFID: 6498-3393)	(UFID: 1553-1845)	(UFID: 5684-4925)

Abstract

Multiple Sequence Alignments forms the basis for phylogenetic analysis about ancestral relationships, helps identify functional sites and detect motifs. There are many methods which solve multiple sequence alignment problem and Center star method is one of them. However, the algorithm requires running time of k^2n^2, where n is length of each k sequence, which makes it difficult to use it for aligning large scale data. The main idea of this paper is to provide an implementation for the improvised version of Center star algorithm using concepts like keyword tree and sliding window for aligning matched regions. The algorithm uses Global Alignment (Needleman Wunsch - dynamic programming) algorithm only to align mismatched sequences, thereby saving a lot of time as compared to aligning entire sequence using dynamic programming.

Introduction

Multiple Sequence Alignment(MSA) Technique is used to align three or more biological sequences. MSA problems can be easily solved by repetitive application of many pairwise sequence alignments, however these naive methods are not adaptive for large number of sequences as it requires $O(2^kn^k)$ time complexity.

DNA sequences are highly homologous. Sequences have a high similarity when they have many matching r-length regions called motifs. These r-length motifs are the substrings patterns which appear in all the sequences. We will take the advantage of these motifs to improvise on the existing Center star algorithm. The center star method is inefficient mainly because of the dynamic programming overhead. The new method focuses on identifying the mismatching r-length regions between the sequences. Only for these mismatched regions, dynamic programming is used for alignment. The algorithm achieves the sequence alignment in two steps: Determination of reference sequence and the sequence alignment. To identify the reference sequence, a keyword tree and sliding window mechanism on the tree can be used. The sequence alignment is done using the reference sequence and the identified mismatching regions found in the sequences. The mismatched regions are then aligned using Global Alignment Strategy (Needleman Wunsch - dynamic programming), followed by center-star alignment to get the result of the MSA.

In this paper, we discuss the results and performance of this improvised version to solve DNA Multiple Sequence Alignment Method as specified in the paper.

Methodology

Phase 1: Reference Sequence Determination -

The first part of the algorithm focuses on the determination of the reference sequence from the given set of sequences. The reference sequence is used for the multiple sequence alignment in the second phase. The reference sequence can be determined by following the steps mentioned below:

1. Construction of Keyword Tree:

Given a genetic sequence G = {P1, P2,....Pn}, each string Pi in the set is divided into smaller sequences Pi = {pi1, pi2,....pih} where the length of each sequence pi is r. A keyword tree Ki is constructed for each of the subsequence in the set G with the root as '*null'*. Each edge in the tree represents a character in the short sequence pi. A path from the root node to the leaf node represents a short sequence. Thus, all the root to leaf paths represent a sequence in the set Pi. In a keyword tree, sequences having common prefix share the same path from the root. A keyword tree is formed for each such set P. An example of keyword tree for the sequence Pi = { ATAC, CGAA, ATGC, ATAG, CGAT} is given below:

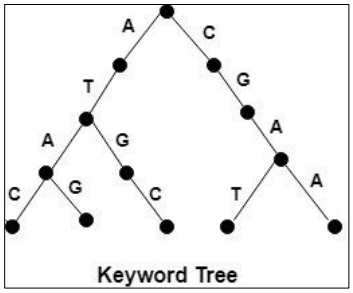

Figure 1: Keyword Tree for the sequence Pi = {ATAC, CGAA, ATGC, ATAG, CGAT}

2. Sliding Window Mechanism

Once the keyword tree is built, then for every sequence Pj in G where $i \neq j$, sliding window of size r is used to find the subsequences that are an exact match with a path of the keyword tree. If a path from root to leaf exactly matches the substring in the window, then the position of this match, both in the sequence Pi and sequence Pj, is recorded. At the same time, a frequency count of the total number of matches of the root-to-leaf path and the substring in Pj is stored. For every mismatch of a character in the sequence Pj, the sliding window is moved by a position equal to the size of the matched part until the current position. The search for matching region again starts from the root of the tree for the new characters in the window.

3. Identification of mismatched regions and determination of the reference sequence

In our implementation of the algorithm, sliding window routine creates and returns a data structure that stores the list of start positions of the matching r-length regions and also stores the number (frequency) of distinct r-length matches between the 2 sequences under consideration.

Thus for a given sequence, when compared with all other sequences, we add up the frequency of distinct matches (referred as total frequency) .

This process is repeated for each of the sequences in the database and we keep a track of the maximum total frequency encountered so far. The sequence for which the maximum value occurs is selected as the reference sequence. The reference sequence has the maximum count of distinct r-length matches with all other sequences in the database.

Phase 2: Sequence Alignment

1. Global Alignment for mismatched regions

From phase 1, we now have the reference sequence that is to be pairwise aligned with all other sequences in the database. We have already stored the start positions of the matching r-length regions corresponding to the pair of sequence under consideration as stated in phase 1.

With this information, the mismatching regions in the pair of sequences can be identified and aligned using Needleman Wunsch- Dynamic Programming algorithm. The following picture demonstrates the working of pairwise alignment algorithm.

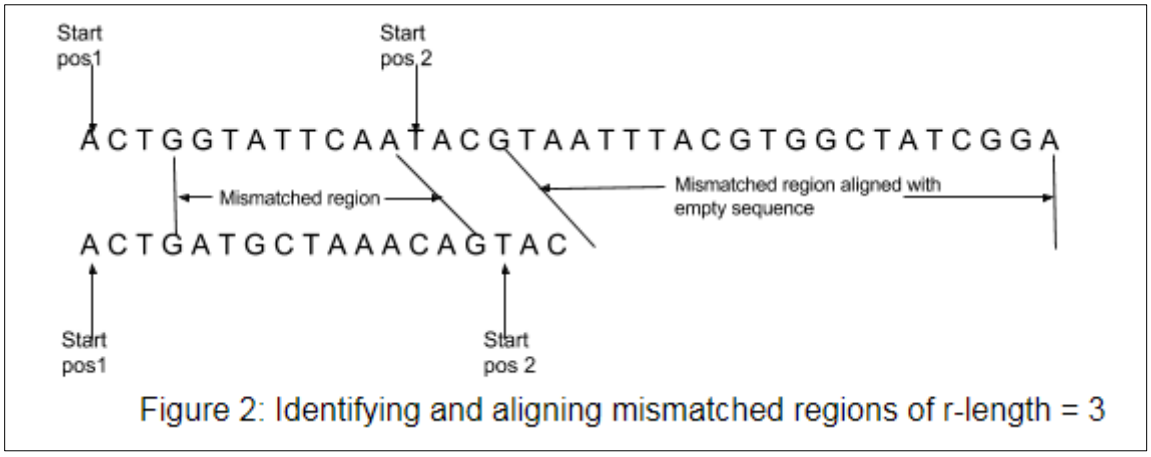

Figure 2: Identifying and aligning mismatched regions of r-length = 3

In figure 2, (start pos 1) and (start pos 2) are the start positions for matching r-length regions in the pair of sequence. The region between the indices (start pos 2) and ((start pos 1) + r) is the mismatched region and is aligned using dynamic programming algorithm.

2. Center star alignment for final MSA

The reference string and the pairwise alignment of all the sequences with respect to the reference string is used for performing the multiple sequence alignment. While creating a pairwise alignment, we keep track of the string with maximum length, say *'str'*.

In this step, we first create a 2D matrix of size *'number of input sequences'* row and '*length of string str'* columns and fill it with '-'. Next, we first populate the matrix with positions of each sequences based on its alignment with respect to reference string.

In phase 1, we recorded the position of each matching r-length regions instead r-length regions itself. Our pairwise alignment algorithm would also append positions instead of value at that index while giving out alignment for mismatched r-length regions.

The reference string is the first entry in 2-D matrix, the second entry will be for string '*str*' with maximum length, subsequent entries will be populated from the original input dataset after

comparing the positions of the reference string based on their respective alignment with reference string in first entry of 2-D matrix. For example:

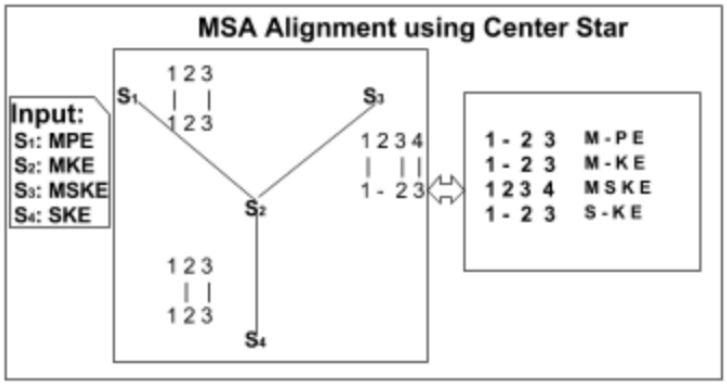

Figure 3: Center Star MSA

Once we have populated it with positions, we simply replace these positions by actual value by extracting characters from respective string in that index to get the multiple sequence alignment of all the sequences in the database.

Results

Complexity Analysis:

Let k be the number of sequences and n be the average length of all the sequences

Phase 1: Reference Sequence Determination

1. Keyword Tree Generation: O(n)
2. Sliding Window on the keyword tree: O(k * n * number of mismatches between a pair of sequences)

Overall Complexity Phase 1: O((k2) * n * number of mismatches between a pair of sequences)

Phase 2: Global Sequence Alignment and Center Star Alignment

1. Global Sequence Alignment: O(n2) -> This is usually less than O(n2) because we do not align the entire pair of sequences using DP, instead we only align the mismatching regions with DP
2. After improvement, Center Star Alignment requires: O(k*n)

Overall Complexity Phase 2: O(k * (n2))

Total Complexity: O((k2) * n * number of mismatched sequences) + O(k * (n2))

System Configuration:

The implementation of the algorithm was done on the UF CISE machines having the configuration:

64-bit Intel Xeon E3 with 3 GHz processor frequency

Test Data:

The data set used for measuring the performance of the implemented algorithm can be downloaded from the site:

ftp://ftp.ebi.ac.uk/pub/databases/fastafiles/emblcds/em_cds_con_hum.gz

The dataset contains many DNA sequences in FASTA format. The selected dataset is a constructed using set of human genome sequences and it has highly similar patterns useful in identifying motifs or ancestral relations thereby making it ideal candidate for our project.

Performance comparison with the normal center star alignment method:

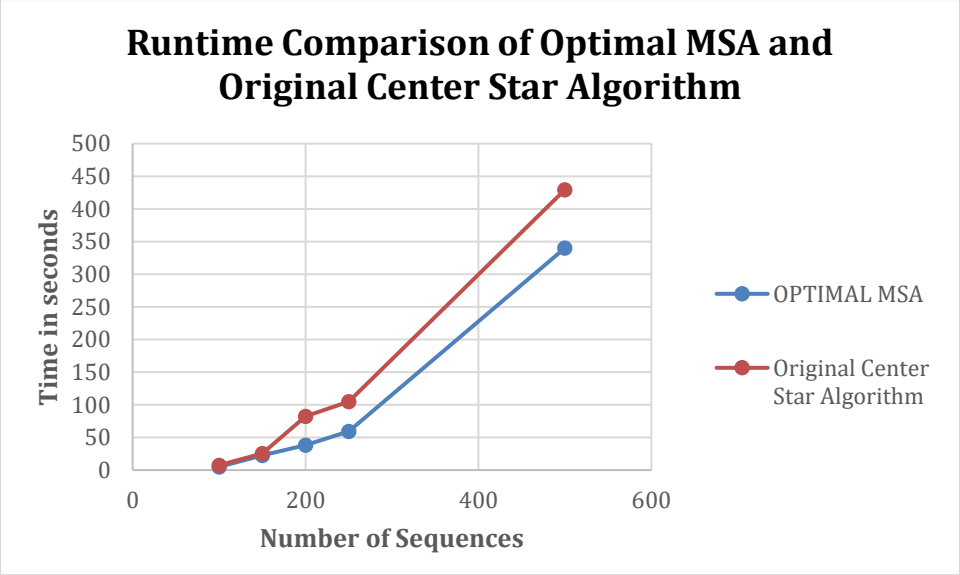

Figure 4: Runtime comparison of Optimal MSA and Original Center Star Algorithm

Number of sequences	Optimal MSA(in ms)	Original Center Star(in ms)
100	4902	7421
150	22673	25555
200	38190	82313
250	59339	105033
500	340087	429387

Table 1: Performance of dataset on Optimal MSA and Original Center Star Algorithms

Table 1 shows the results of the comparison between Optimal MSA and Original Center Star Algorithm. Experimental units of the running time are all millisecond (ms).

For comparison between the runtime of Optimal MSA and original Center Star Algorithm, we divided the dataset into chunks of smaller sequences each of size k and we observed that the Optimal MSA algorithm significantly improves runtime complexity against the original Center Star Algorithm by eliminating application of dynamic programming on entire sequences as compared to original method.

Conclusion

From the above implementation and analysis of the results we conclude that the optimal algorithm reduces the runtime of the original Center Star Algorithm by using keyword tree and sliding window to find the matching substrings and then only use Dynamic Programming algorithm to align the mismatched regions. For the sequences with highly repetitive motifs, the algorithm would be able to align many matching r-length regions with the sliding window and thereby leave very little overhead on dynamic programming.

Workload Distribution

1. Develop understanding of MSA and reviewing the optimized algorithm as mentioned in the paper[1]- Jinal, Poornima, Krishit
2. Designing the overall process and the different modules needed for the optimized multiple sequence alignments - Jinal, Poornima, Krishit
3. Implementation of the node structure for the keyword tree and construction of the Keyword tree for all the sequences in the database. - Poornima
4. Implementation of Sliding Window to identify the matching regions between two sequences - Jinal
5. Reference sequence determination and pairwise alignment of mismatched regions - Krishit
6. Merging the results obtained in the above steps to compute the center star alignment for MSA - Jinal, Poornima, Krishit
7. Comparison and evaluation of performance measures of the optimized implementation of MSA as compared to the original MSA - Jinal, Poornima, Krishit

Github Repository

The implementation of the algorithm can be found on the Github repository:
https://github.com/jmashruw/Optimal-Center-Star

References

1. Yong Sun, Zili Zhang, and Jun Wang , "A Novel Algorithm for DNA Multiple Sequence Alignment Based on the Sliding Window and the Keyword Tree", International Journal of Bioscience, Biochemistry and Bioinformatics, Vol. 3, No. 3, May 2013
2. S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence alignment of two proteins,” Journal of Molecular Biology, vol. 48, pp. 443-453, 1970
3. D. Gusfield, “Algorithms on strings, trees and sequences: Computer Science and Computational Biology,” Cambridge England, Cambridge University Press, 1997, pp. 505-523