CS575: Final Project Report

Project Title: DNA Sequence Alignment

Team Member(s): Yash Patel , Pranav Sahu

I. PROBLEM

Sequence matching and alignment plays a vital role in computational biology. The functional and structural similarity can be determined by the regions of similarity between two sequences. Given two or more sequences find the region of similarity and dissimilarity in those sequence in a way that either they create the perfect match or best possible alignment. For this, we have studied and implemented different kind of string matching and string alignment algorithms and calculated their time and space complexity.

II. ALGORITHMS

A. *Brute-force for string match*

Brute force algorithm is probably the simplest algorithm. Here we start by comparing the first character of each sequence, if it is a match then we continue matching the second character and so on otherwise we shift one of the sequences (preferably smaller one) and again start by comparing the newly aligned characters we keep on doing this until a perfect match is found.

B. *Knuth–Morris–Pratt (KMP)*

The idea behind the KMP algorithm is to avoid the redundant comparisons that were done in the brute force approach. It uses degenerating property of pattern and improve the worst case time complexity to O(n). Degenerating property means patterns having the same sub-patterns appearing more than once in the pattern are considered. Here we preprocess the array by constructing an lps[] (longest proper prefix suffix) array corresponding to pattern string of the same size as pattern string. At any iteration lps[i] stores the length of the maximum matching proper prefix which is also a suffix of pattern p[0..i]. By storing this information it goes through each substring only once, means there is no retracting in the array and this results in time complexity of O(m+n)

C. *Needleman Wunsch*

This is a dynamic programming approach to the problem. Here we aim to globally align the sequences rather than perfectly matching them. First, we create a score matrix. The scoring scheme used is as follows, match(+1), mismatch(-1), gap (-2). Using the scoring scheme we fill up the values in the score matrix by following a few simple rules.

1. *Initialize the bottommost right element to zero and the first character of both strings as a gap.*
2. *If the value comes from the bottom or right element we simply add the gap(-2)*
3. *If the value comes from the diagonal element we either consider it as a match (+1)or a mismatch(-1).*

After creating the score matrix we do a trace back and finally we align the sequences. Aligning the sequences is also governed by some rules.

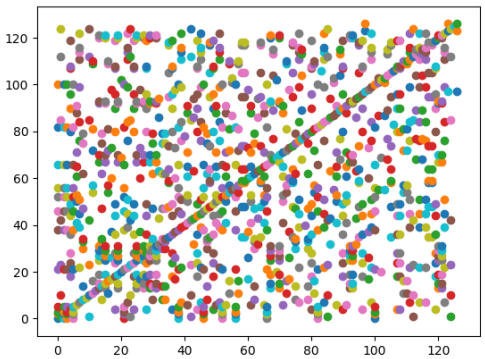
1. *Start from the element in the matrix which has the highest value (this value corresponds to the highest score)*
2. *Trace back to the starting point, that is to the bottom-most left element by using tracing arrows in score matrix*
3. *In order to align the two sequences we put characters from both strings, if a traceback comes from a diagonal "D" or we put a gap if the trace back comes from the bottom "U"or left "R".*

D. *Hybrid Algorithm*

This algorithm uses both recursion (topdown) and Dynamic programming (bottom-up) approach to solve the sequence alignment problem. In this algorithm, we divide the longer sequence into smaller subsequences recursively until a pre-defined size is reached (let us say k). Now we apply the perfect string matching algorithm such as brute force or Knuth-Morris-Pratt (KMP) to those smaller subsequences, which will reduce the data size by a significant amount and at last perform the string alignment on the rest of the subsequences. Then construct the solution by concatenating the locally aligned sequences. The efficiency of the algorithm can be tested by comparing the final score to the score we get with the dynamic programming approach. This gives a rough estimate of how well the substrings are aligned, which in our case is the within the error margin of 5%.

E. *Dot Plot*

Dot plot can be used to infer the rough alignment information of the smaller protein sequences. This approach simply takes into account the match and mismatch between two strings of same length, which is often found in protein sequences. When there is a character match we place 1 in the matrix or else we put a zero for a mismatch, then we make a plot based on this matrix to see what is the span of the plot. This plot can be used to quickly infer what kind of protein sequences are being compared.



In above graph the diagonal shows that the major components of the protein sequence matches. The sequences used to plot this are Transthyretin D and Transthyretin B respectively .

III. Software Design and Implementation

A. *Software Design*

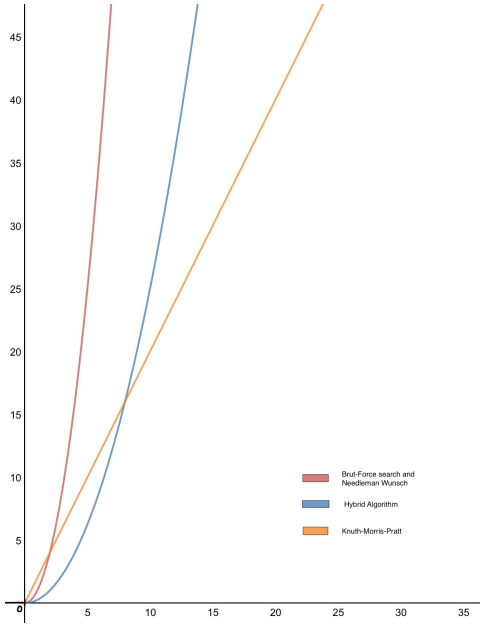
The software design is done in python.

B. *Implementation and Tools Used*

Brute-Force search, Needleman Wunsch, Hybrid Algorithm is being implemented from the scratch along with the function for creating the dot plot. Tools used are numpy and matplotlib for reading and processing the sequence and plotting the results respectively.

C. *Performance Evaluation*

Based on runtime (latency) of the algorithm the brute force approach takes the **5 minutes 40 seconds** in order to align a string of 100000 characters long. The result is **4 minutes 49 seconds** for the Needleman Wunsch Algorithm for the sequence of the same length as both of the algorithms are of O(m*n). The hybrid approach we think will produce better results in term of latency. Apart from this, the hybrid algorithm can process larger sequences without any buffer over flow. The hybrid algorithm managed to give the output in **28 seconds**, which is a huge performance boost as compared to the previous two algorithms.



The above graph essentially shows that the brute force search and Needleman-Wunsch Algorithm for the string alignment takes $O(n^2)$. The hybrid is also a $O(n^2)$ time algorithm but the speedup can be seen by the slope of the curve while the Knuth-Morris-Pratt algorithm take O(m+n) time. The actual results resonates with the expected output.

Attachments

- Source code (attached to this report)
- Git link : https://github.com/yashpatel007/DNA-Sequence-Alignment-DAA-project.git

References

1. D.Gusfield, Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, 1997.
2. P. Manohar, Shailendra Singh, Protein Sequence Alignment: A Review World Applied Programming, Vol (2), No (3), March 2012. 141-145.
3. Algorithms for computational biology : MIT OCW Course 6-096 spring 2005. Link : https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-algorithms-for-computational-biology-spring-2005/lecture-notes/lecture5_newest.pdf