A New DNA sequencing alignment methodology using the Longest Common Subsequence technique

Debkumar Chowdhury* and Kartik Sau
University of Engineering & Management, Kolkata, India

Sanjukta Mishra
Brainware University, Kolkata, India

The goal of this paper is to perform DNA sequence alignment using the Longest Common Subsequence Algorithm to obtain the longest common subsequence from the two strings X and Y for matching DNA in molecular biology etc. This paper would seek to establish the differences between the 2 species DNA and thereby conclude on their genetic similarity. The paper will be implemented in terms of space and time using the optimized longest common subsequence algorithm. This paper acts as a bridge between computer science and genomics, exploring the interdisciplinary area of computational biology and helping us to consider the very fundamental concepts of life and our relationship with other species.

Keywords: Computational Biology, Genetic Engineering, Medical Diagnosis, Bioinformatics.

I. INTRODUCTION

DNA, or deoxyribonucleic acid, is the human genetic stuff, and virtually all other species. In a persons body, virtually every cell has the same DNA. Much DNA is in the nucleus of the cells (where it is called nuclear DNA), but in the mitochondria, there is also a significant amount of DNA (where it is called mitochondrial DNA or mtDNA). In cells, mitochondria are structures that transform food energy into a form that cells can utilize.

DNA data is stored as a file consisting of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). DNA of a person is composed of approximately 3 billion bases, and more than 99

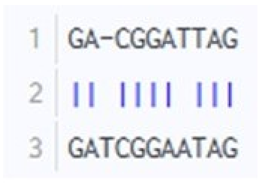
One essential role in molecular biology is to compare two sequences, known as sequence comparison, either from the same organism or from a separate organism. It helps to provide solutions to many biological problems, such as:

- Predicting the structure and function of proteins.
- Inferring evolutionary history and species relatedness.
- Identifying specific subsequences in genes/proteins to classify specific motifs, as a sub-problem in DNA sequencing genome assembly mutating the genomes to reproduce a whole species.

II. SEQUENCE ALIGNMENT

Sequence alignment is a way of putting the correspondence between related characters or substrings one series over series. This may occur in deoxyribonucleic acid (DNA), ribonucleic acid (RNA), or protein sequences. Sequences of various species can have varying sizes. Alignment involves the addition of spaces in the series in random places so that both are of the same size. Spaces or gaps are either added at the start or the end of the series.

Lets take an example here.



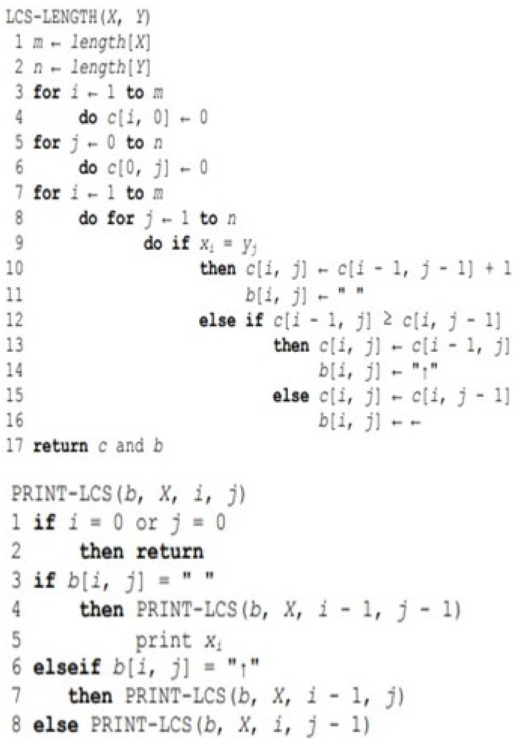
```
GA-CGGATTAG
|| |||| |||
GATCGGAATAG
```

FIG. 1: Base Comparison.

III. LITERATURE SURVEY

Several methods had proposed throughout all the years to perform DNA sequencing alignment using LCS. They suffer from limited advantages and major drawbacks. We have tried to highlight some of them and tried to project their core methodologies in this section. H. Rick [1] in 1955 introduced an algorithm based on advancing from contour to contour which was compared with other existing algorithms and proved to be better among them. Wagner and Fischer [1] in the year 1974 introduced an algorithm using the concept of a matrix to get the solution for this problem with dynamic programming. This algorithm just gives the LCS length but not the LCS. Fully use the divide and conquer technique and complex method, In 1975 Hirschberg [1] developed a method for finding LCS. The dominant match was another approach given by Hirschberg [1] in 1977. J.W. Hunt, T.G. [1]

*Email: debkumar.cse@gmail.com



```
LCS-LENGTH(X, Y)
m ← length[X]
n ← length[Y]
for i ← 1 to m
    do c[i, 0] ← 0
for j ← 0 to n
    do c[0, j] ← 0
for i ← 1 to m
    do for j ← 1 to n
           do if x_i = y_j
                 then c[i, j] ← c[i - 1, j - 1] + 1
                      b[i, j] ← " "
                 else if c[i - 1, j] ≥ c[i, j - 1]
                         then c[i, j] ← c[i - 1, j]
                              b[i, j] ← "↑"
                         else c[i, j] ← c[i, j - 1]
                              b[i, j] ← ←
return c and b

PRINT-LCS(b, X, i, j)
if i = 0 or j = 0
    then return
if b[i, j] = " "
    then PRINT-LCS(b, X, i - 1, j - 1)
         print x_i
elseif b[i, j] = "↑"
   then PRINT-LCS(b, X, i - 1, j)
else PRINT-LCS(b, X, i, j - 1)
```

FIG. 2: Description of the algorithm.

in 1977, Szymanskis claimed that computing LCS from two strings is tantamount to finding the longest monotonically increasing path in the graph, where $x_i = y_j$. Another algorithm was published by Apostolico, A. & Guerrain [1] in 1987, which was an alternative to support the framing of the LCS. In the year 1990, Wu [1] made an effort to decrease the issue of editing distance to reduce edit distance and to apply it to finding LCS. The time complexity of this algorithm is O (n (m-r)). Elham Parvinnia, M. Taheri, Koorush Ziarati [2] proposed an Improved Longest Common Subsequence Algorithm for Reducing Memory Complexity in Global Alignment of DNA Sequences in 2008. Sergey Sheetlin, Yonil Park, and John L. Spouge [3] in 2011 proposed an objective method for estimating asymptotic parameters, with an application to sequence alignment. Their publicly available computer program ARRP replaces the subjective assessment of the asymptotic regime with an objective change-point detection method, increasing confidence in the scientific objectivity of the parameter estimates. Izzat Alsmadi and Maryam Nuser [4] proposed a string matching evaluation method for DNA comparison in 2012. Youhei Namiki, Takashi Ishida, and Yutaka Akiyama [5] proposed a fast DNA Sequence Clustering Based on Longest Common Subsequence in 2013. The method is called LCS-HIT based on the popular CD-HIT program. The proposed method employs a novel filtering technique based on the longest common subsequence to identify similar sequence pairs. This filtering technique affords a considerable speed-up over CD-HIT without loss of sensitivity. Spouge J L, Mario-Ramrez L, Sheetlin SL [6] proposed the generalized Ruzzo-Tompa algorithm to find optimal subsequences with gaps in 2014 where the linear-time Ruzzo-Tompa (RT) algorithm finds subsequences of unusual composition, using a sequence of scores as input and the corresponding 'maximal segments' as output. Murugan and U. Udayakumar [7] proposed the sequence similarity between genetic codes using improved longest common subsequence algorithm in 2017. Deena Nath, Jitendra Kurmi, and Deveki Nandan Shukla [8] proposed a Revised Algorithm to find the Longest Common Subsequence in 2018.

IV. PROPOSED METHODOLOGY

A. Longest Common Subsequence (Nave Approach)

Find the length of the longest subsequence present in both sequences, given two parts. A subsequence is a string that exists but not necessarily contiguous in the same relative order. “abc”,“bdf”,“aeg”, “acefg”, “bdf”, “bdf”,“bdf” .Subsequences of “abcdefg” are, etc.

For determiningthe difficulty of the brute force method, the number of differing Length subsequences of a string n must first be identified. Remember from permutation theory and the combination that numerous combinations with 1 variable are nC1. The combination number of 2 elements is nC2 and so on. ${}^nC_0 +^n C_1 +^n$

$C_2 + ...^nC_n = 2^n$. A string of length n, therefore, has 2^n-1 separate potential subsequences, since we do not find the subsequence of length 0. This means that this approachs time complexity would be O (n * 2^n). Note it has O (n) time to search for both strings to have a subsequence common. With DP this runtime complexity can be decreased.

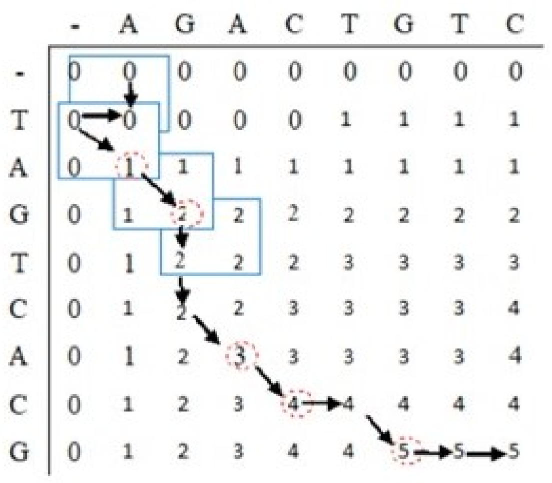


FIG. 3: Tracing path in Longest Common Subsequence.

B. Algorithm

The simplistic approach to this problem would be to generate all the subsequences of the two components and also to seek the resulting longest subsequence. This solution is exponential in terms of time complexity. Let us see how this problem possesses both important properties of a Dynamic Programming (DP) problem.

C. Optimal Substructure (Dynamic Programming solution)

Suppose we have two Sl and S2 sequences of lengths m and n respectively, where S1= $a_1, a_2, .., a_n$, and $S2 = b_l, b_2, .b_n$. Well create a matrix A where A_{ij} denotes the length of $a_la_2......a_i$ and $b_lb_2..b_j$ the longest common subsequence.

Sequence S1 is written vertically, and S2 is written horizontally. Compare every symbol expressing the rows to the column expressing every letter. We must continue Row by Row, Column by Column. 1. If $a_i = b_j$, we did consider a match. For the current match, we get a score of 1 and from the rest of the LCS, weve already received substrings a_1 all and $b_1b_{(j-l)}$

$$A_{i,j} = \begin{cases} A_{i-1,j-1} + 1 \text{ if } a_i = b_j \\ max(A_{i-1,j}, A_{i,j-1}) + 1 \text{ if } a_i \neq b_j \end{cases}$$

It takes O (nm) time to fill in the m by n matrix A. This approach is called dynamic programming.

D. Flowchart

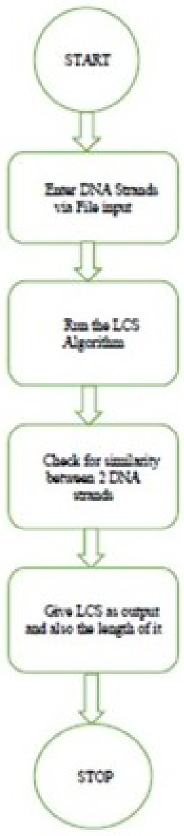

START
Enter DNA Strands via File input
Run the LCS Algorithm
Check for similarity between 2 DNA strands
Give LCS as output and also the length of it
STOP

FIG. 4: Flowchart of the proposed algorithm.

V. EXPERIMENTAL RESULTS AND ANALYSIS

All the works related to our paper, have been carried out in C++ language using TDM-GCC 4.9.2 64-bit profiling compiler. The IDE used is Dev C++. The computation machine used is HP Pavilion AU 003tx.

A. Proposed approach vs. Brute force approach

To evaluate the complexity of the brute force solution, we must first know the number of possible different remainders from permutation theory and the combination that numerous combinations with 1 variable are nC1. The combination number of 2 elements is nC2 and so on. nC0 + nC1 + nC2 + nCn = 2n. A string of length n, therefore, has 2n-1 separate potential subsequences, since we do not find the subsequence of length 0. This means that the brute force approachs time complexity would be O (n * 2n). Note it takes O (n) time to search for both strings to have a subsequence common. With our proposed approach this runtime complexity can be decreased. With the help of Dynamic Programming, the time complexity is O(m.n). We can see how this method has helped DNA sequencing instead of just normal brute forcing method and thereby decrease the overall time complexity.

B. Graph Comparison

In the below graph we display a graph showing the previous DNA sequencing alignment method using the longest common subsequence utilizing Brute Force strategy. In the below graph we display a graph showing our

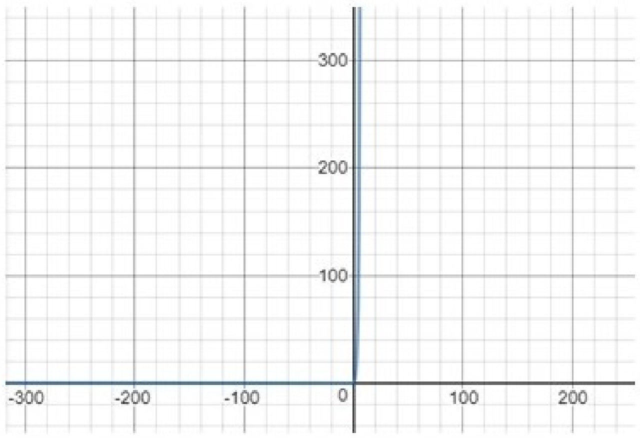


FIG. 5: Graph showing the previous DNA sequencing alignment method using the longest common subsequence utilizing Brute Force strategy.

proposed new DNA sequencing alignment method using the longest common subsequence utilizing dynamic programming strategy.

VI. CONCLUSION

In this paper, we have shown how the Longest Common Subsequence algorithm can be used to compare 2 DNA strands and thereby showing its importance in interdisciplinary fields. We have used the Dynamic Programming approach for our proposed work which reduces the complexity to a great extent and thereby is more efficient than other methods based on recursive or brute force approach. With the help of experimental results and analysis, we have demonstrated its performance as compared to other nave approaches. This algorithmic approach can also be used to detect the protein sequence of an unknown virus and compare its similarity with other

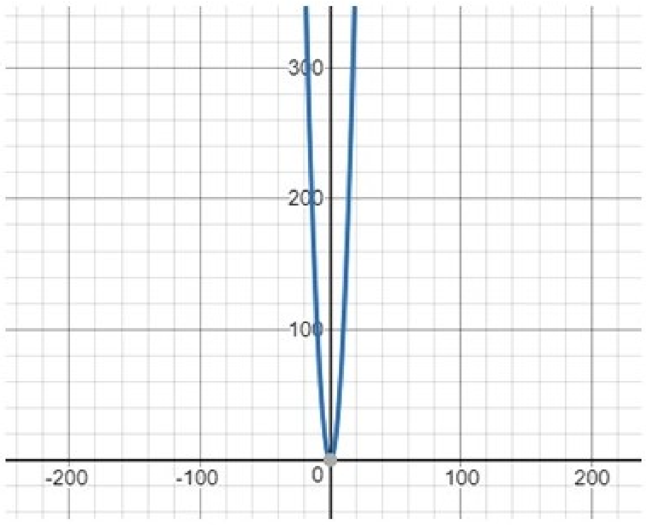


FIG. 6: Graph showing a new DNA sequencing alignment method using the longest common subsequence utilizing dynamic programming strategy

viruses and thereby increasing the chances to find its similarity with other known viruses for which we already have a vaccine.

[1] Deena Nath, Jitendra Kurmi, and Vipin Rawat, "A Survey on Longest Common Subsequence", International Journal for Research in Applied Science & Engineering Technology (IJRASET), and ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887, Volume 6 Issue IV, April 2018.

[2] Parvinnia, Elham & Taheri, M. & Ziarati, Koorush. (2008). An Improved Longest Common Subsequence Algorithm for Reducing Memory Complexity in Global Alignment of DNA Sequences. 57-61. 10.1109/BMEI.2008.212.

[3] Sheetlin, Sergey & Park, Yonil & Spouge, John. (2011). Objective method for estimating asymptotic parameters, with an application to sequence alignment. Physical review. E, Statistical, nonlinear, and soft matter physics. 84. 031914. 10.1103/PhysRevE.84.031914.

[4] Alsmadi, Izzat & Nuser, Maryam. (2012). String Matching Evaluation Methods for DNA Comparison. 47. 13-32.

[5] Namiki, Youhei & Ishida, Takashi & Akiyama, Yutaka. (2013). Acceleration of sequence clustering using longest common subsequence filtering. BMC Bioinformatics. 14. S7. 10.1186/1471-2105-14-S8-S7.

[6] Spouge JL, Mario-Ramrez L, Sheetlin SL, "Searching for repeats, as an example of using the generalized Ruzzo-Tompa algorithm to find optimal subsequences with gaps", Int. J. Bioinformatics Research and Applications, Vol. 10, Nos. 4/5, 2014.

[7] A. Murugan, U. Udayakumar, "Sequence Similarity be-

tween Genetic Codes using Improved Longest Common Subsequence Algorithm", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 Volume: 5 Issue: 7.

[8] Deena Nath, Jitendra Kurmi, Deveki Nandan Shukla, "A Revised Algorithm to find Longest Common Subsequence", International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887, Volume 6 Issue IV, April 2018