COMPARATIVE ANALYSIS OF DYNAMIC PROGRAMMING ALGORITHMS TO FIND SIMILARITY IN GENE SEQUENCES

Shankar Biradar[1], Vinod Desai[2], Basavaraj Madagouda[3], Manjunath Patil[4]

[1, 2, 3, 4] *Assistant Professor, Department of Computer Science & Engineering, Angadi Institute of Technology and Management, Belgaum, Karnataka, India.*

shankar_pda@yahoo.com, vinod.cd0891@gmail.com, basavarajmadagoda@gmail.com, manjunath.patil03@gmail.com

Abstract

There exist many computational methods for finding similarity in gene sequence, finding suitable methods that gives optimal similarity is difficult task. Objective of this project is to find an appropriate method to compute similarity in gene/protein sequence, both within the families and across the families. Many dynamic programming algorithms like Levenshtein edit distance; Longest Common Subsequence and Smith-waterman have used dynamic programming approach to find similarities between two sequences. But none of the method mentioned above have used real benchmark data sets. They have only used dynamic programming algorithms for synthetic data. We proposed a new method to compute similarity. The performance of the proposed algorithm is evaluated using number of data sets from various families, and similarity value is calculated both within the family and across the families. A comparative analysis and time complexity of the proposed method reveal that Smith-waterman approach is appropriate method when gene/protein sequence belongs to same family and Longest Common Subsequence is best suited when sequence belong to two different families.

Keywords *- Bioinformatics, Gene, Gene Sequencing, Edit distance, String Similarity.*

----------***----------

1. INTRODUCTION

Bioinformatics is the application of computer technology to the management of biological information. The field of bioinformatics has gained widespread popularity largely due to efforts such as the genome projects, which have produced lot of biological sequence data for analysis. This has led to the development and improvement of many computational techniques for making inference in biology and medicine. A gene is a molecular unit of heredity of a living organism. It is a name given to some stretches of DNA and RNA that code for a polypeptide or for an RNA chain that has a function in the organism. Genes hold the information to build and maintain an organism's cells and pass genetic characteristic to their child. Gene sequencing can be used to gain important information on genes, genetic variation and gene function for biological and medical studies [13]. Edit distance is a method of finding similarity between gene/protein sequences by finding dissimilarity between two sequences [5]. Edit distance between source and target string is represented by how many fundamental operation are required to transfer source string into target, these fundamental operations are insertion, deletion and subtraction. The similarity of two strings is the minimum number of edit distance. String Similarity is quantitative term that shows degree of commonality or difference between two comparative sequences [10], Finding the gene similarity has massive use in the field of bioinformatics.

2. MATERIALS AND METHODS

In this section we describe the various materials and methods which are used in our algorithms

2.1 Dataset Used

For the experiment purpose we took data sets from 5 different families which are listed below, and the source of information is [16] [17].
Family: kruppel c2h2-type zinc finger protein.
Family: caution-diffusion facilitator (CDF) transporter family.
Family: E3 ubquitin-protein ligase.
Family: Semaphorin-7A.
Family: SPAG11 family.

2.2 Dataset Format

In this research work we used various data sets from different families for the implementation of different algorithms, all this data set is in FASTA format. In bioinformatics, FASTA format is a text-based format for representing nucleotide sequences, in which nucleotides or amino acids are represented using single-letter codes. The format also contain sequence name before the sequences start. A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. The word following the ">" symbol is the identifier of the sequence, and the rest of the line is the

description (both are optional). There should be no space between the ">" and the first letter of the identifier. It is recommended that all lines of text be shorter than 80 characters. The sequence ends if another line starting with a ">" appears; this indicates the start of another sequence.

2.3 Gap Penalty

In order to get best possible sequence alignment between two DNA sequences, it important to insert gaps in sequence alignment and use gap penalties. While aligning DNA sequences, a positive score is assigned for matches negative score is assigned for mismatch To find out score for matches and mismatches in alignments of proteins, it is necessary to know how often one sequence is substituted for another in related proteins. In addition, a method is needed to account for insertions and deletions that sometimes appear in related DNA or protein sequences. To accommodate such sequence variations, gaps that appear in sequence alignments are given a negative penalty score reflecting the fact that they are not expected to occur very often. It is very difficult to get the best-possible alignment, either global or local, unless gaps are included in the alignment.

2.4 Blosum Matrix

A Blosum matrix is necessary for pair wise sequence alignment. The four DNA bases are of two types, purines (A and G) and pyrimidines (T and C). The purines are chemically similar to each other and the pyrimidines are chemically similar to each other. Therefore, we will penalize substitutions between a purine and a purine or between a pyrimidine and a pyrimidine (transitions) less heavily than substitutions between purines and pyrimidines (transfusions). We will use the following matrix for substitutions and matching's. The score is 2 for a match, 1 for a purine with a purine or a pyrimidine with a pyrimidine, and -1 for a purine with a pyrimidine.

3. ALGORITHMS

Dynamic programming algorithms for finding gene sequence similarity are discussed in detail in this section along with pseudo codes and algorithms. We used three algorithms for analysis purpose, all these algorithms uses the concept of dynamic programming, which is output sequence depends upon the input of previous sequence. Those three algorithms are.

a. Levenshtein edit distance algorithm
b. Longest common subsequence algorithm
c. Smith-waterman algorithm

3.1 Levenshtein Edit Distance Algorithms

It is one of the most popular algorithms to find dissimilarity between two nucleotide sequences, it is an approximate string matching algorithm mainly used for forensic data set, the basic principle of this algorithm is to measure the similarity between two strings [4]. This is done by calculating the number of basic operations as mentioned in introduction part. Algorithm for Levenshtein edit distance is as fallows

```
Int LevenshteinDistance (char S[1..N], char T[1...M] )
    {
    Declare int D[1....N, 1....M]
    For i from 0 to N
    D[i,0] := i // the distance of any first string to an empty
    second string
    For j from 0 to M
    D[0, j] := j // the distance of any second string to an empty
    first string
    For j from 1 to M
        {
      For i from 1 to N
          {
            S[i] = T[j] then
            D[i, j] := D[i-1, j-1]
            Else
            D[i,j] := min {
                    D[i-1 , j] +1 // deletion
                    D[i, j-1] +1 // insertion
                    D[i-1, j-1] +1// substitution
                            }
          }
        }
 Return D[M, N]
    }
```

3.2 Longest Common Subsequence Algorithm

Finding LCS [3] [8] is one way of computing how similar two sequences are, Longer the LCS more similar they are. The LCS problem is a special case of the edit distance problem. LCS is similar to Levenshtein edit distance algorithm except few steps and it also involves trace back process in order to find similar sequences.

Algorithm for Longest common subsequence is as fallows

```
Int LCS (char S[1...N], char T[1....M] )
{
Declare int D[1....N, 1....M]
For i from 0 to N
D[i,0] := 0
For j from 0 to M
D[0, j] := 0
For j from 1 to M
    {
  For i from 1 to N
      {
   D[i,j] := max {
                V1;
                V2;
                V3+1 if S=T else V3
```

```
                    }
            }
        }
Return D [M, N]
 }
```

Where V1 = the value in the cell to the left of current cell.
V2= the value in the cell above the current cell.
V3= value in the cell above left to the current cell,
S and T are source string and Target string respectively

3.3 Smith-Waterman Algorithm

The Smith–Waterman algorithm is a well-known algorithm for performing local sequence alignment; that is, for determining similar regions between two nucleotide or protein sequences. Instead of looking at the total sequence, the Smith–Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure.

Smith-waterman algorithm differ from other Local alignment algorithm in fallowing factors

a. A negative score/weight must be given to mismatches.
b. Zero must be the minimum score recorded in the matrix.
c. The beginning and end of an optimal path may be found anywhere in the matrix not just the last row or column.

Pseudo code core smith-waterman algorithm is as fallows.

```
Pseudo code for initialization of matrix
For i=0 to length(A)
F(i,0) ← d*i
For j=0 to length(B)
F(0,j) ← d*j
For i=1 to length(A)
For j=1 to length(B)
{
        Diag ← F(i-1,j-1) + S(Ai, Bj)
        Up ← F(i-1, j) + d
        Left ← F(i, j-1) + d
        F(i,j) ← max(Match, Insert, Delete)
}
Pseudo code for SW alignment
For (int i=1;i<=n;i++)
For (int j=1;j<=m;j++)
    int s=score[seq1.charAt(i-1)][seq2.charAt(j-1)];
    int val=max(0,F[i-1][j-1]+s,F[i-1][j]-d,F[i][j-1]-d);
    F[i][j]=val;
    If (val==0)
        B[i][j]=null;
      Else if(val==F[i-1][j-1]+s)
        B[i][j]=new Traceback2(i-1,j-1);
      Else if(val==F[i-1][j]-d)
        S[i][j]= new Traceback2(i-1,j);
      Else if(val==F]i][j-1]-d)
        B[i][j]= new Traceback2(i,j-1);
```

Where i and j are columns and rows respectively, S (xi; yj) is value of substitution matrix and g is gap penalty, the substitution matrix is a matrix which describes the rate at which one character in a sequence changes to other character states over time

3.4 Results within the Same Families

Table1. Family: cation-duffusion facilitator

Seq1 protein name	Seq2 protein name	Seq1 length	Seq2 length	Similarity in LCS	similarity in LD	Similarity in SW
Zinc transporter 9	CDF family transporter1	577	313	159	132	266
Zinc transporter 9	CDF family transporter2	577	309	161	138	288
Zinc transporter 9	CDF	577	312	150	125	241
CDF family transporter1	CDF family transporter2	311	309	157	91	284
CDF family transporter1	CDF	311	312	142	95	269
CDF family transporter2	CDF	309	312	135	87	282

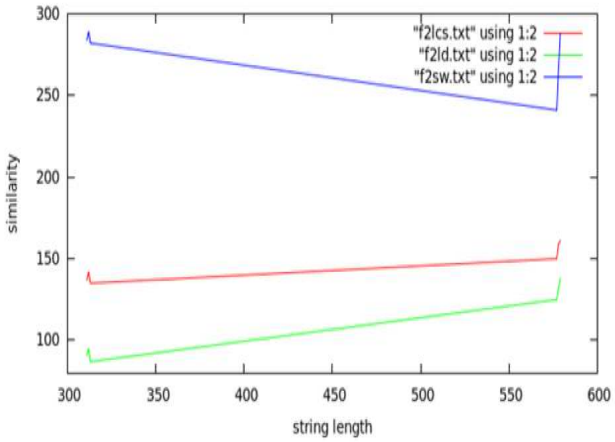

Figure1. Similarity graph for cation-duffusion facilitator family

Table2. Family: semaphorin

Seq1: protein name	Seq2: protein name	Seq1: Length	Seq2: Length	Similarity in LCS	Similarity in LD	Similarity in SW
Semaphorin-7A	Semaphorin-4C	728	849	287	174	568
Semaphorin-7A	Semaphorin-3A	728	786	287	167	598
Semaphorin-7A	Semaphorin-3E	728	833	303	214	495
Semaphorin-7A	Semaphorin-4B	728	838	306	219	602
Semaphorin-7A	Semaphorin-4A	728	774	291	169	592

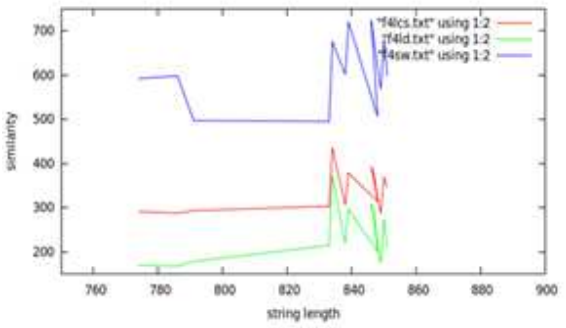

Figure2. Similarity graph for family semaphorin

In the figure 2, blue line indicates similarity in smith-waterman algorithm, red line indicate similarity in longest common subsequence algorithm and finally the green line indicate similarity in Levenshtein algorithm. As we see from the graph smith-water man algorithm is more efficient then other two algorithms while finding the similarity of gene sequences that belonging to same family.

3.5 Results between Different Families

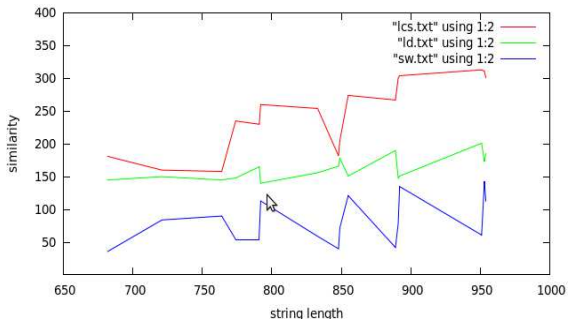

Figure3. Similarity graph across family

Where red line indicates LCS algorithm, green line indicates Levenshtein edit distance algorithm and blue line is for smith-waterman algorithm. From the above graph, we can conclude that while comparing two gene sequences belonging to different families, longest common subsequence is better algorithm because it gives maximum similarity as compare to other two algorithms.

Table3. Similarities across the families

Seq1:protein name	Seq2:protein name	Seq1: length	Seq2:length	Similarity in LCS	similarity in LD	Similarity in SW
Zinc finger protein 224	CDF family transporter	721	313	160	150	84
Zinc finger protein 234	CDF family transporter	764	309	158	145	90
Zinc finger protein 234	E3ubiquitin-protein ligase UPL5	764	951	313	201	61
Zinc finger protein 120	Semaphorin-4C	446	849	205	178	72
Zinc finger protein 120	Sperm-associated antigen 11	446	111	76	80	63
CDF family transporter 1	E3ubiquitin-protein ligase UPL5	572	951	251	202	101
CDF family transporter	E3ubiquitin-protein ligase Itchy	339	889	176	167	83
CDF family transporter 1	Semaphorin-4F	572	791	230	165	53
CDF family transporter	Semaphorin-4B	309	848	182	166	40
CDF family transporter 1	Semaphorin-4A	428	774	235	148	53

CONCLUSIONS

We considered finding the gene sequence similarity using dynamic programming for our project work. In dynamic programming there exist many different approaches to find similarity among gene sequences; we took some of these algorithms for our project and did comparative analysis of these algorithms using datasets from five different families. We took different protein sequences from all these dataset as input to our program and did rigorous experimentation on these datasets, both within the families and across the families. Five data sets which are used for our experimental work are kruppel c2h2-type zinc finger protein, cation-diffusion facilitator (CDF) transporter, E3 ubquitin –protein ligase, semaphorin and finally SPAG11B and got the results as discussed in the previous section. From the results we can conclude that smith-waterman algorithm is best suited to find similarity for protein sequences that belonging to the same family, and longest common subsequence algorithm is best

suited for protein sequences that are belong to different families.

REFERENCES

[1] S. Hirschberg; "Algorithms for the longest common subsequence problem". J.ACM, 24;{664-675};1977

[2] Levenshtein V.I; "binary code capable of correcting deletion, insertion and reversal"; soviet physics doklady; vol 8; 1966

[3] Ristead, R.S Yianilos,P.N; "learning string edit distance"; IEEE Transaction or pattern analysis and machine intelligence; 1998

[4] L. Bergroth; "Survey of Longest Common Subsequence Algorithms"; Department of Computer Science, University of Turku20520 Turku,Finland; 2000 IEEE

[5] Hekki Hyyro, Ayumi Shinohara; "A new bit-parallel-distance algorithm"; Nikoltseas.LNCS 3772; 2005

[6] Adrian Horia Dediu,et al; "A fast longest common subsequence algorithm for similar strings"; Language and automation theory and application, International Conference, LATA; 2010

[7] Patsaraporn Somboonsat, Mud-Armeen munlin; "a new edit distance method for finding similarity in DNA sequence"; world academy of science engineering and technology 58; 2011

[8] Dekang Lin; " An Information-Theoretic Definition of Similarity"; Department of Computer Science University of Manitoba,Winnipeg, Manitoba, Canada R3T 2N2

[9] Xingqin Qi, Qin Wu, Yusen Zhang2, Eddie Fuller and Cun-Quan Zhang1; "A Novel Model for DNA Sequence Similarity Analysis Based on Graph Theory"; Department of Mathematics,

[10] West Virginia University, Morgantown, WV, USA, 26506. School of Mathematics and Statistics,Shandong University at Weihai, Weihai, China, 264209 Gina M. Cannarozzi; "String Alignment using Dynamic Programming"

[11] David R Bentley; Whole-genome re-sequencing.

[12] M. Madan Babu; Biological Databases and Protein Sequence Analysis; Center for Biotechnology, Anna University, Chenna

[13] A pattern classification; Richard O.Duda, peter E.Hart, David G.Stork 2nd editon;

[14] simultaneous solution of the RNA folding alignment and protosequence problem; David Sankoff Siam ,J.Apple Math; vol 45; 1985

[15] http://www.ncbi.nlm.nih.gov/

[16] http://www.uniprot.org/uniprot/