X

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΙΑΤΡΙΚΗΣ
ΠΜΣ «Μεθοδολογία Βιοϊατρικής Έρευνας, Βιοστατιστική και Κλινική Βιοπληροφορική»

COMPARISON OF THE NEEDLEMAN – WUNSCH ALGORITHM AND THE SMITH WATERMAN ALGORITHM FOR ALIGNING PAIRS OF PROTEIN SEQUENCES

ΣΥΓΚΡΙΣΗ ΤΩΝ ΑΛΓΟΡΙΘΜΩΝ NEEDLEMAN – WUNSCH ΚΑΙ SMITH WATERMAN ΓΙΑ ΤΗΝ ΕΥΘΥΓΡΑΜΜΙΣΗ ΤΩΝ ΖΕΥΓΑΡΙΩΝ ΣΤΙΣ ΠΡΩΤΕΪΝΙΚΕΣ ΑΛΛΗΛΟΥΧΙΕΣ

ΜΠΕΤΑ ΧΡΙΣΤΙΝΑ

16 ΙΟΥΝΙΟΥ 2017

ΤΡΙΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

ΑΡΤΕΜΙΣ ΧΑΤΖΗΓΕΩΡΓΙΟΥ ΗΛΙΑΣ ΖΙΝΤΖΑΡΑΣ ΧΡΥΣΟΥΛΑ ΔΟΞΑΝΗ

ABSTRACT

Bioinformatics, a new scientific discipline, addresses the specific needs in data acquisition, storage, analysis and integration that research in genomics generates. Because of the rapidly increase in size of databases of protein sequences, it becomes more and more reliable to depend on classified proteins to determine the structure and function of a novel protein. One way to build a sequence alignment is to find a series of insertions, deletions, or replacements, collectively called mutation events, which will transform one sequence into another. Sequence alignment algorithms allow the matching of a residue with a gap element or simply a gap. For this reason we analyze and compare global and local alignment algorithms in this project. Global alignment is a form of global optimization that "forces" the alignment to span the entire length of all query sequences and local alignments identify regions of similarity within long sequences that are often widely divergent overall. Contrary to the Needleman-Wunsch algorithm, on which the Smith-Waterman algorithm is built, the Smith-Waterman algorithm is searching for local alignments, not global alignments, considering segments of all possible lengths to optimize the similarity measure. The Smith-Waterman algorithm is becoming more and more widespread, is mathematically rigorous, thus it is guaranteed to find the best scoring alignment between the pair of sequences being compared. Both of them produce optimal alignment between the sequences. So if you want to know which algorithm to use you have to decide if you want to compare closely related sequences of same length or identify regions of similarity within long sequences that are often widely divergent overall.

I. INTRODUCTION

Genomic technologies are generating an extraordinary amount of information, unprecedented in the history of Biology. Thus, a new scientific discipline, Bioinformatics, at the intersection between Biology and Computation, has recently emerged. Bioinformatics addresses the specific needs in data acquisition, storage, analysis and integration that research in genomics generates. The most important aspect in bioinformatics at the molecular level is sequence alignment providing an explicit mapping between two or more biological sequences such as DNA and proteins.

DNA (or deoxyribonucleic acid) is the molecule that carries the genetic information and consists of chemical and physical components. It belongs to a class of molecules called the nucleic acids, which are polynucleotides – that is, long chains of nucleotides. DNA is composed of two DNA strands each one consists of nitrogenous bases as cytosine (C), guanine (G), adenine (A), or thymine (T). Genes consist of specific sequences of nucleotides at a given position on a given chromosome that code for a specific protein. A protein is a linear polymer of 20 different types of amino acids, linked together by peptide bonds. Chemical properties that distinguish the 20 different amino acids cause the protein chains to fold into three-dimensional structures that define functions in the cell. Databases of protein sequences has increased in size, thus it becomes more and more reliable to depend on classified proteins to determine the structure and function of a novel protein.

II. METHODS

In Bioinformatics, a sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural or evolutionary relationships between the sequences. A sequence alignment is a residue by residue matching between two sequences and algorithms that look for homologous regions between two sequences by aligning them residue by residue are indisputably the most fundamental components of sequence comparison. It is also biologically relevant to consider situations where nucleotides have either been inserted into or deleted from DNA.

Most, but not all, sequence alignment algorithms allow the matching of a residue with a gap element or simply a gap. Consider two sequences which are identical except that the first sequence contains one extra residue. When we view the alignment of these two sequences, the extra residue will be matched to a gap. This corresponds to an insertion event in the first sequence or a deletion event in the second. On the other hand, if we note that an insertion event has occurred in the first sequence (with respect to the second) then we know how to match that residue to a gap in the second. Thus one way to build a sequence alignment is to find a series of insertions, deletions, or replacements, collectively called mutation events, which will transform one sequence into another. The number of mutation events needed to transform one sequence into another is called the edit distance. As there will always be more than one series of possible mutation events which transform the first sequence into the second, it makes sense to rate each set's likelihood of occurrence. Greater confidence is placed in alignments which have a higher likelihood of occurring. Each alignment can be rated by considering both the cumulative probabilities and biological significance of each mutation event. For example, an alignment which infers a lesser amount of mutations to transform one sequence into another is almost always considered more likely to have occurred than an alignment which

infers many more mutations, therefore many alignment algorithms work by minimizing the edit distance. This information is most commonly stored, in a scoring matrix. Each pair of matched residues (or residue – gap pairs) can be scored and the similarity score is the sum of the scores of the individual residues.

Most alignment algorithms seek to produce meaningful alignments by maximizing the similarity score for two sequences. Traditionally, sequence alignment algorithms have been called global if they seek to optimize the overall alignment of two sequences. Often the resulting alignment can include long stretches of residues which are matched with different residues or gaps. Conversely, if the algorithm seeks to align highly conserved subsequences while ignoring any intervening unconserved regions then it is called a local alignment algorithm. A local alignment of two sequences can produce a number of different subsequent alignments.

Pair wise Sequence Alignment

Pair wise sequence alignment is one of the most fundamental methods for comparing DNA and protein sequences. It establishes the basis for the interpretation of evolutionary and functional relationships between gene sequences and species. Pair wise alignment aligns two sequences at one time whereas multiple sequence alignment method aligns more than two sequences at a time. Such comparison involves aligning sections of the two sequences in a way that exposes the similarities between them. Pair wise sequence alignment methods are used to find the best-matching piecewise (local) or global alignments of two query sequences. They can only be used between two sequences at a time, but they are efficient to calculate and are often used for methods that do not require extreme precision (such as searching a database for sequences with high similarity to a query). The three primary methods of producing pair wise alignments are dot-

matrix methods, dynamic programming, and word methods; however, multiple sequence alignment techniques can also align pairs of sequences. Although each method has its individual strengths and weaknesses, all three pair wise methods have difficulty with highly repetitive sequences of low information content - especially where the number of repetitions differ in the two sequences to be aligned.

Local – Global Alignment

Very short or very similar sequences can be aligned by hand. However, most interesting problems require the alignment of lengthy, highly variable or extremely numerous sequences that cannot be aligned solely by human effort. Instead, human knowledge is applied in constructing algorithms to produce high-quality sequence alignments, and occasionally in adjusting the final results to reflect patterns that are difficult to represent algorithmically (especially in the case of nucleotide sequences). Computational approaches to sequence alignment generally fall into two categories: *global alignments* and *local alignments*. Calculating a global alignment is a form of global optimization that "forces" the alignment to span the entire length of all query sequences. By contrast, local alignments identify regions of similarity within long sequences that are often widely divergent overall. Local alignments are often preferable, but can be more difficult to calculate because of the additional challenge of identifying the regions of similarity. A variety of computational algorithms have been applied to the sequence alignment problem. These include slow but formally correct methods like dynamic programming. These also include efficient, heuristic algorithms or probabilistic methods designed for large-scale database search that do not guarantee to find best matches.

Global alignment algorithms attempt to find a single overall alignment from one network to another, and local aligners may output several mutually incompatible alignments for the input networks. Local alignment is more useful when we desire to identify several potential orthologs per input protein, whereas

global alignment is more helpful for identifying larger conserved networks that are indicative of a common ancestor. Global alignments can also be somewhat easier to interpret, as the produced mapping is one to one.

EXAMPLE

Global Alignment	Local Alignment
S I M I L A R I T Y	M I L A R
P I – L L A R – – –	I L L A R

Sequence alignment is an important application where dynamic programming is necessary. The technique of dynamic programming can be applied to produce global alignments via the Needleman - Wunsch algorithm, and local alignments via the Smith - Waterman algorithm. In typical usage, protein alignments use a substitution matrix to assign scores to amino-acid matches or mismatches, and a gap penalty for matching an amino acid in one sequence to a gap in the other. DNA and RNA alignments may use a scoring matrix, but in practice often simply assign a positive match score, a negative mismatch score, and a negative gap penalty. (In standard dynamic programming, the score of each amino acid position is independent of the identity of its neighbors, and therefore base stacking effects are not taken into account. However, it is possible to account for such effects by modifying the algorithm.) A common extension to standard linear gap costs is the usage of two different gap penalties for opening and extending a gap.

III. SMITH – WATERMAN ALGORITHM

The Smith-Waterman algorithm is a dynamic programming method for determining similarity between nucleotide or protein sequences. The algorithm was first proposed in 1981 by Smith and Waterman and is identifying homologous regions between sequences by searching for optimal local alignments. To find the optimal local alignment, a scoring system including a set of specified gap penalties is used [Smith and Waterman, 1981]. Homology identified by sequence database searches often implies shared functionality

between sequences and further research and development might depend on the accuracy of the search results. The Smith-Waterman algorithm is built on the idea of comparing segments of all possible lengths between two sequences to identify the best local alignment. This means that the Smith-Waterman search is very sensitive and ensures an optimal alignment of the sequences. Unfortunately, this also has the effect that the method is both time and CPU intensive.

Needleman and Wunsch (1970) were the first to introduce a heuristic alignment algorithm for calculating homology between sequences. Later, a number of variations have been suggested, among others Sellers (1974) getting closer to fulfill the requests of biology by measuring the metric distance between sequences [Smith and Waterman, 1981]. Further development of this led to the Smith-Waterman algorithm based on calculation of local alignments instead of global alignments of the sequences and allowing a consideration of deletions and insertions of arbitrary length.

The Smith-Waterman algorithm is the most accurate algorithm when it comes to search databases for sequence homology but it is also the most time consuming, thus there has been a lot of development and suggestions for optimizations and less time-consuming models. One example is the well-known Basic Local Alignment Search Tool, BLAST [Shpaer et al., 1996].

How does it work?

The Smith-Waterman algorithm is searching for homology by comparing sequences. When sequences are compared using local alignments, the total number of alignments can be considerable, and identification of the best alignments is of high importance to both reliability and relevance of the data obtained. This identification of the optimal local alignment between two sequences is basically what the Smith-Waterman algorithm does. Optimal local alignments are identified by comparing the query sequence and the sequences in the database on a character-to-character level. Contrary to the Needleman-Wunsch algorithm, on which the Smith-Waterman algorithm is built, the Smith-Waterman algorithm is searching for local alignments, not

global alignments, considering segments of all possible lengths to optimize the similarity measure [Smith and Waterman, 1981].

The basis of a Smith-Waterman search is the comparison of two sequences $A = (a_1a_2a_3 \ldots a_n)$ and $B = (b_1b_2b_3 \ldots b_m)$

The Smith-Waterman algorithm uses individual pair-wise comparisons between characters as:

$$H_{ij} = max = \begin{cases} H_{i-1,j-1} + s(a_i, b_j) \\ \max_k\{H_{i-k,j} - W_k\} \\ \max_l\{H_{i,j-l} - W_l\} \\ 0 \end{cases}$$

H_{ij} is the maximum similarity of two segments ending in a_i and b_j respectively.

Similarity of residues a_i and b_j is given by a weight matrix considering match, substitution or insertion/deletion.

First term considers an extension of the alignment by extending the two sequences compared by one residue each.

Second and third term handle an extension of the alignment by inserting a gap of length k into sequence A or sequence B, respectively.

Finally, fourth term placing a zero in the recursion ignores a possible negative alignment score.

Preceding calculations will be started neutral and the allowance of the similarity score to be zero in the expression for H_{ij} means that a local alignment can restart at any position performing a character-to-character comparison.

The algorithm assigns a score to each residue comparison between two sequences. By assigning scores for matches or substitutions and insertions/deletions, the comparison of each pair of characters is weighted into a matrix by calculation every possible path for a given cell. In any matrix cell the value represents the score of the optimal alignment ending at these coordinates and the matrix reports the highest scoring alignment as the optimal alignment (see *Figure 1*). For constructing the optimal local alignment from the matrix, the

starting point is the highest scoring matrix cell. The path is then traced back through the array until a cell scoring zero is met. Because the score in each cell is the maximum possible score for an alignment of any length ending at the coordinates of this specific cell, aligning this highest scoring segment will yield the highest scoring local alignment - the optimal local alignment.

EXAMPLE

Sequence A: CAGCCUCGCUUAG

Sequence B: AAUGCCAUUGACGG

Parameters for the scoring matrix being:

Match = 1

Mismatch = −⅓

Gap = −(1 + ⅓ k), k being the gap extension number.

The similarity matrix is filled as shown in *Figure 1*.

		Sequence A												
		C	A	G	C	C	U	C	G	C	U	U	A	G
	A	0,0	1,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,0
	A	0,0	1,0	0,7	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,7
	U	0,0	0,0	0,8	0,3	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,7
	G	0,0	0,0	1,0	0,3	0,0	0,0	0,7	1,0	0,0	0,0	0,7	0,7	1,0
	C	1,0	0,0	0,0	2,0	1,3	0,3	1,0	0,3	2,0	0,7	0,3	0,3	0,3
	C	1,0	0,7	0,0	1,0	3,0	1,7	?						
Sequence B	A													
	U													
	U													
	G													
	A													
	C													
	G													
	G													

Figure 1 : Calculation of the similarity matrix considering penalties for gap initiations and extensions. Values are assigned to each cell based on the parameter settings. Adapted from [McLysaght,].

As any cell value represents the score of the optimal alignment ending at the cell coordinates, the highest scoring position in the matrix reports the ending point of the highest scoring and thereby the optimal alignment between the two sequences compared. To construct the optimal alignment, the starting point is the cell with the highest scoring value representing the last residue in this alignment. The complete alignment is identified by tracing back through the array from this highest scoring matrix cell until a cell scoring zero is met. Illustrated in Figure 2, the highest scoring cell is identified with the score of 3.3 and is traced back six steps. The search for local alignments allowing any position to be starting point and any position to be ending point means that the optimal alignment can be of any possible length and is thereby identified as the optimal local alignment.

		Sequence A												
		C	A	G	C	C	U	C	G	C	U	U	A	G
	A	0,0	1,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,0
	A	0,0	1,0	0,7	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,7
	U	0,0	0,0	0,8	0,3	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,7
	G	0,0	0,0	1,0	0,3	0,0	0,0	0,7	1,0	0,0	0,0	0,7	0,7	1,0
	C	1,0	0,0	0,0	2,0	1,3	0,3	1,0	0,3	2,0	0,7	0,3	0,3	0,3
	C	1,0	0,7	0,0	1,0	3,0	1,7	1,3	1,0	1,3	1,7	0,3	0,0	0,0
Sequence B	A	0,0	2,0	0,7	0,3	1,7	2,7	1,3	1,0	0,7	1,0	1,3	1,3	0,0
	U	0,0	0,7	1,7	0,3	1,3	2,7	2,3	1,0	0,7	1,7	2,0	1,0	1,0
	U	0,0	0,3	0,3	1,3	1,0	2,3	2,3	2,0	0,7	1,7	2,7	1,7	1,0
	G	0,0	0,0	1,3	0,0	1,0	1,0	2,0	3,3	2,0	1,7	1,3	2,3	2,7
	A	0,0	1,0	0,0	1,0	0,3	0,7	0,7	2,0	3,0	1,7	1,3	2,3	2,0
	C	1,0	0,0	0,7	1,0	2,0	0,7	1,7	1,7	3,0	2,7	1,3	1,0	2,0
	G	0,0	0,7	1,0	0,3	0,7	1,7	0,3	2,7	1,7	2,7	2,3	1,0	2,0
	G	0,0	0,0	1,7	0,7	0,3	0,3	1,3	1,3	2,3	1,3	2,3	2,0	2,0

Figure 2: Identification of optimal local alignment from similarity matrix. To identify the optimal local alignment comparing two sequences according to the Smith-Waterman algorithm, the highest scoring cell in the similarity matrix is identified. As any cell value represents the value of local alignments of arbitrary length ending at these specific coordinates, back tracing from the highest scoring cell leads to the highest scoring alignment - the optimal alignment. Adapted from [McLysaght,].

The alignment represented by the path shown in red in the similarity matrix in *Figure 2* is:

Sequence B: G C C A U U G

Sequence A: G C C - U C G

IV. NEEDLEMAN – WUNSCH ALGORITHM

The Needleman Wunsch Algorithm performs a global alignment on two sequences. It is sometimes referred as optimal matching algorithm. This algorithm provides a method of finding the optimal global alignment of two sequences by maximizing the number of amino acid matches and minimizing the number of gaps. In pair wise sequence alignment algorithms, a scoring function must exist such that different scores can be assigned to different alignments of two proteins relative to the number of gaps and number of matches in the alignment. Thus, the alignment with the largest score must be the optimal alignment.

The Needleman-Wunsch algorithm consists of three steps:

1. Initialization of the score matrix
2. Calculation of scores and filling the trace back matrix
3. Deducing the alignment from the trace back matrix

How does it work?

This is a highly accurate, dynamic programming based, global pair wise alignment algorithm. It was originally developed for aligning protein sequences but can also be used to align DNA sequences. This algorithm aligns two sequences A and B with lengths m and n residues respectively, by finding a path through a two dimensional m×n array; C. As all m×n values in C must be calculated for every alignment, the work needed to align two sequences becomes intractable for large m and n. First the bottom right cell $C_{m,n}$ is assigned the value 1 or 0 depending on whether the base in position m of A matches the base in position n of B. The cell diagonally above and to the left of this cell; $C_{m-1, n-1}$, is given a value of 1 if the base in position m – 1 of A matches the base in position n – 1 of B or a value of -1 otherwise. This is because a match will produce a maximum run of matches of length 2 for all bases from this point on, whereas a mismatch will produce a run of at most one match. The algorithm continues working backwards until every cell has been assigned a value. Finally the algorithm starts from the highest scoring cell in the array, and finds a path

through the array which maximizes the cumulative sum of the values in the cells visited in the path. The resulting path represents a maximally matching global alignment.

A matrix *D(i,j)* indexed by residues of each sequence is built recursively, such that:

$$D(i, j) = max\begin{cases} D(i-1, j-1) + s(x_i, y_i) \\ D(i-1, j) + g \\ D(i, j-1) + g \end{cases}$$

Subject to a boundary conditions. *s(i, j)* is the substitution score for residues *i* and *j*, and *g* is the gap penalty

The cells of the score matrix are labeled *C (i, j)* where *i = 1, 2, …, m* and *j = 1, 2, …, n*

The score of any cell *C(i, j)* is the maximum of:

$q_{diag} = C(i - 1, j - 1) + S(i, j)$

$q_{up} = C(i - 1, j) + g$

$q_{left} = C(i, j - 1) + g$

where *S(i,j)* is the substitution score for letters *i* and *j*, and *g* is the gap penalty.

EXAMPLE

Sequence A: G C A T G C U

Sequence B: G A T T A C A

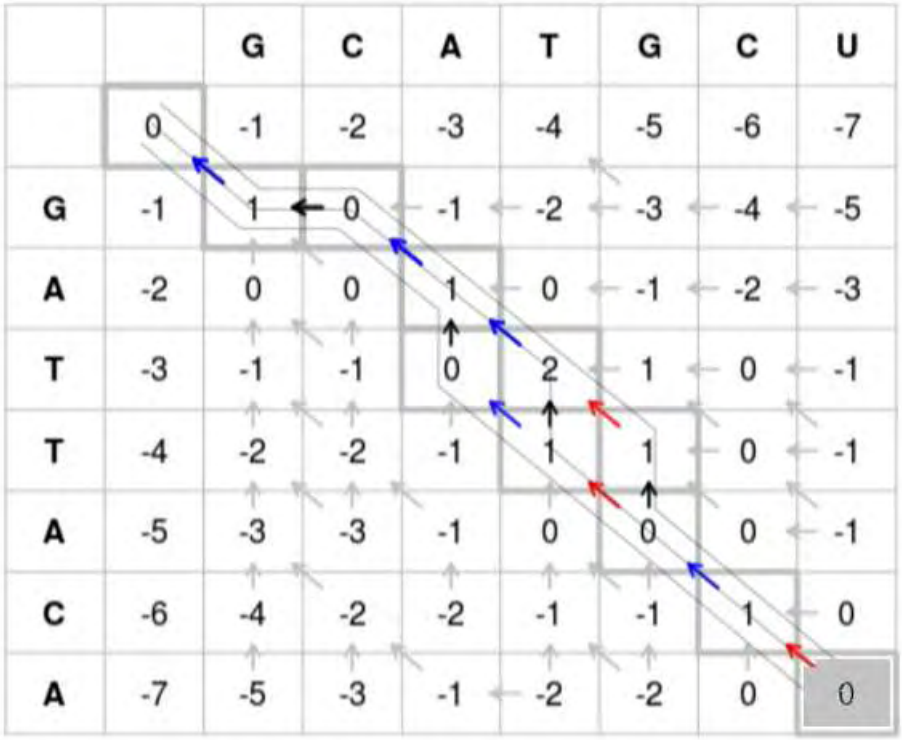
Parameters for the scoring matrix being:

Match = 1

Mismatch = −1

Gap = −1.

The similarity matrix is filled as shown in *Figure 3*



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

Figure 3: Calculation of the similarity matrix considering penalties for gap initiations and extensions. Identification of optimal global alignment from similarity matrix. To identify the optimal global alignment comparing two sequences according to the Needleman-Wunsch agorithm, the highest scoring cell in the similarity matrix is identified.

Best alignments as shown in *Figure 3:*

- Sequence A: G C A T G − C U

  Sequence B: G − A T T A C A

- Sequence A: G C A − T G C U

  Sequence B: G − A T T A C A

- Sequence A: G C A T − G C U

  Sequence B: G − A T T A C A

V. RESULTS

The Smith-Waterman algorithm is quite time demanding because of the search for optimal local alignments, and it also imposes some requirements on the computer's memory resources as the comparison takes place on a character-to-character basis. The fact that similar searches using the Smith-Waterman algorithm take a lot of time often prevents this from being the first choice, even though it is the most precise algorithm for identifying homologous regions between sequences. Today's research requires fast and effective data analysis. Other faster methods like BLAST or FASTA have therefore largely replaced Smith-Waterman searches as demands to time of handling large amounts of data are still getting stronger. On the other hand, large-scale projects are getting more and more prevalent and the researchers are becoming more and more concerned about the risk of missing important information if not using the most sensitive algorithm for database searches. As a consequence, the use of the Smith-Waterman algorithm is again becoming more and more widespread. Smith – Waterman is mathematically rigorous, thus it is guaranteed to find the best scoring alignment between the pair of sequences being compared. As one of local alignment methods is more useful for dissimilar sequence that is suspected to contain regions of similarity or similar sequence motifs within their large sequence. Instead of aligning the entire length of two protein sequences, this algorithm finds the region of highest similarity between to proteins. This is potentially more biologically relevant due to the fact that the ends of proteins tend to be less highly conserved than the middle portions, leading to higher mutation, deletion and insertion rates at the ends of the protein. The Smith – Waterman algorithm allows us to align proteins more accurately without having to align the ends of related protein which may be highly different.

On the other hand, Needleman – Wunsch algorithm tries to achieve the best global alignment, over the entire input. Ii is suitable for two sequences with similar length and significant degree of overall similarity. The

algorithm is carried out from beginning till end of the sequence to find out the best possible alignment. The Needleman–Wunsch algorithm is still widely used for optimal global alignment, particularly when the quality of the global alignment is of the utmost importance. However, the algorithm is expensive with respect to time and space, proportional to the product of the length of two sequences and hence is not suitable for long sequences.

VI. CONCLUSION

In this paper we have tried to display the two dynamic programming algorithms and their application in biological sequence alignment. To align two sequences, the Needleman – Wunsch algorithm produces optimal global alignment with gaps and the Smith – Waterman algorithm will locally align sub sequences. Both of these algorithm use dynamic programming to optimize the scores based on substitution matrices. We focused on explaining how each algorithm works and tried to compare them. They achieve the same goal – alignment but optimize for different criteria. The basic difference is that Needleman Wunsch algorithm tries to achieve the best global alignment that can be applied in two sequences which are of the similar length and similar across their entire lengths. On the other hand if we want to align highly conserved subsequences while ignoring any intervening unconserved regions then we are using Smith – Waterman algorithm. It is also a bit more useful for tasks like determining the difference between DNA sequences, because usually researchers are more interested in the shift in the sequence of the gene, allowing them to determine the divergence over time. Both of them produce optimal alignment between the sequences. So if you want to know which algorithm to use you have to decide if you want to compare closely related sequences of same length or identify regions of similarity within long sequences that are often widely divergent overall.

VII. REFERENCES

1. Abhilash C B. , Rohitaksha K. (2014) : A Comparative Study on Global and Local Alignment .Algorithm Methods
2. Sudha M.P , P. Sripriya (2014) : Sequence Alignment in DNA Using Smith –Waterman and Needleman Algorithms.
3. Marc A. Marti-Renom , M.S. Madhusudhan and Andrej Sali (2004) : Alignment of Protein Sequences by their Profiles.
4. Connor Clark , Jugal Kalita (2014) : A Comparison of Algorithms for the Pairwise Alignment of Biological Networks.
5. Gina M. Camarozzi (2011) : String Alignment Using Dynamic Progrmming
6. Stephen F. Altschul : Global and Local Sequence Alignment , National Center for Biotechnology Information
7. Yasuyki Nozaki and Mathew Bellgard (2004) : Centre for Bioinformatics and Biological Computing, Murdoch University, Murdoch, WAGISO, Australia
8. Needleman S.B. and Wunsch C.D. (1970) : A General Method Application to the Search for Similarities in the Amino Acid Sequence of Two Proteins. Journal of Molecular Biology
9. Smith T.F. and Waterman M.S. (1981) : Identification of Common Molecular Subsequences. Journal of Molecular Biology