Jurnal Teknologi

Full Paper

# REDUCING THE SEARCH SPACE AND TIME COMPLEXITY OF NEEDLEMAN-WUNSCH ALGORITHM (GLOBAL ALIGNMENT) AND SMITH-WATERMAN ALGORITHM (LOCAL ALIGNMENT) FOR DNA SEQUENCE ALIGNMENT

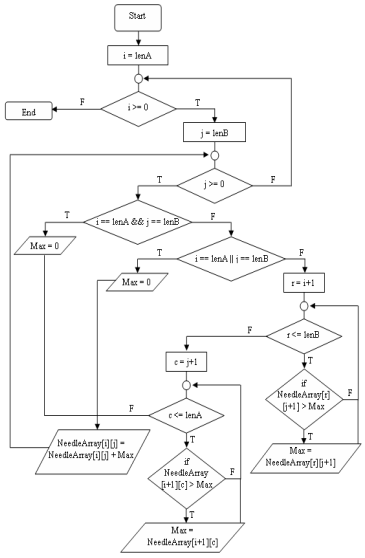
F. N. Muhamad[a*], R. B. Ahmad[a], S. Mohd. Asi[a], M. N. Murad[b]

[a]School of Computer & Communication Engineering, Universiti Malaysia Perlis, Perlis, Malaysia
[b]School of Manufacturing Engineering, Universiti Malaysia Perlis, Perlis, Malaysia

**Article history**
Received
*15 May 2015*
Received in revised form
*1 July 2015*
Accepted
*11 August 2015*

*Corresponding author
fatimah_noni@yahoo.com

**Graphical abstract**




## Abstract

The fundamental procedure of analyzing sequence content is sequence comparison. Sequence comparison can be defined as the problem of finding which parts of the sequences are similar and which parts are different, namely comparing two sequences to identify similarities and differences between them. A typical approach to solve this problem is to find a good and reasonable alignment between the two sequences. The main research in this project is to align the DNA sequences by using the Needleman-Wunsch algorithm for global alignment and Smith-Waterman algorithm for local alignment based on the Dynamic Programming algorithm. The Dynamic Programming Algorithm is guaranteed to find optimal alignment by exploring all possible alignments and choosing the best through the scoring and traceback techniques. The algorithms proposed and evaluated are to reduce the gaps in aligning sequences as well as the length of the sequences aligned without compromising the quality or correctness of results. In order to verify the accuracy and consistency of measurements obtained in Needleman-Wunsch and Smith-Waterman algorithms the data is compared with Emboss (global) and Emboss (local) with 600 strands test data.

*Keywords*: Sequence comparison, sequence alignment, global alignment, local alignment, similarity, optimal alignment, scoring, traceback, match, mismatch, gaps



## 1.0 INTRODUCTION

The data representation in this work is DNA sequences. DNA in a genome is not neatly arranged and stores an organism's genetic information in the form of a long sequence of molecules. Specifically, the information is encoded using four key chemicals, *adenine*, *thymine*, *guanine* and *cytosine* (abbreviated as A, T, G and C) [1]. The DNA sequences of hundreds of organisms have been decoded and stored in databases. This biological sequence data can be obtained from variety of public and private databases.

With the growing amount of data, it became impractical to analyze DNA sequences manually, so faster algorithms and tools are needed. Sequence analysis is the process used to find information about a *nucleotide* or *amino acid* sequence using computational methods [2]. The fundamental procedure of analyzing sequence content is sequence comparison. Sequence comparison is the cornerstone of Bioinformatics. Sequence comparison

is regarded as one of the most fundamental problems of computational biology, which is usually solved with a technique known as sequence alignment. Sequence alignment can be defined as the problem of finding which parts of the sequences are similar and which parts are different. Generally, it is the process of comparing two sequences to identify similarities and differences between them. So, a measure of how similar they are is also desirable, then, a typical approach to solve this problem is to find a good and plausible alignment between the two sequences.

The field of bioinformatics consists of many computationally challenging problems, many of which involve very complex system. Sequence alignment is a problem of paramount importance and is a fundamental operation performed in computational biology research. The goal is to produce the best alignment for a pair of DNA or protein sequences (represented as strings of characters). A good alignment has zero or more gaps inserted into the sequences to maximize the number of positions in the aligned strings that match. For example, consider aligning the sequences "ATTGGC" and "AGGAC". By inserting gaps ("-") in the appropriate place, the number of positions where the two sequences agree can be maximized: "ATTGG-C" and "A–GGAC" [3]. The other main problem in sequence alignment is making sequences having the same size with the insertion of gaps in locations along the sequences and creating a correspondence between sequences. The sequence alignment problem can be illustrated as, given a scoring function that measures the score of aligning characters at the same position from each sequence, calculate the total score of the alignment by adding the scores of all positions and find the maximum total score of every possible alignments. The effort is still being studied by researchers in finding the best way to align sequences by reducing the gaps and sizes of the sequences.

One of the problems in the comparison of sequences of biological data is an effort to determine their degree of similarity. So, many available algorithms and techniques in solving the problems of sequence alignment. There are many algorithms that maximize speed and do not concern with the accuracy of the result alignment. And also, there are many algorithms that maximize accuracy and do not concern with the speed. Most current sequence comparison methods used in practice, such as, BLAST [4] and FASTA [5] are based on Heuristics [6] which are much faster, but do not provide optimal results.

There are many algorithms written that use the approach of Dynamic Programming. However, Needleman-Wunsch algorithm was the first to introduce Dynamic Programming to compare biological sequences for finding the *global* alignment between two sequences [7]. Later, the improvement from Needleman-Wunsch algorithm proposed Smith-Waterman algorithm to find the best *local* alignment between two sequences [8].

In this work, the study on how to analyze large sequences and to reduce the search space and time complexity without compromising the accuracy and efficiency is presented. This is by evaluating the performance of Needleman-Wunsch and Smith-Waterman algorithms in finding the optimal alignment between a pair of DNA sequences. However, this work only focus on similarity, as it is the preferred choice for biological applications.

### 1.1 Sequence Alignment Algorithm

Once the alignments are constructed, the next step is to determine which ones scored the highest points. Given a *scoring matrix*, these optimal alignments can be determined using existing algorithms written and use the approach of Dynamic Programming technique, depending on the desired comparison. The Needleman-Wunsch algorithm computes optimal global alignments, while the Smith-Waterman algorithm computes local alignments.

Global sequence alignment refers to the process of finding the best possible alignment between two given sequences by considering the sequences [9]. It is comparing the sequences entirely, which attempt to align every letter in every sequences, are most useful when the sequences in the query set are similar and of roughly equal size. The Needleman-Wunsch 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* necessary to align the two sequences [10]. *Gaps* can be introduced in between any of the sequences because it can produce a better alignment. *Gaps* are insertion and deletion of base pairs and are commonly occurring changes in sequences aligned. The similarity score is the score of the best alignment by adding up all the scores of the *matches* and the *mismatches*. *Gaps* are usually given a negative score along with *mismatches*, while *matche*s produce a positive score.

From the observation, basically, the concept behind the Needleman-Wunsch algorithm that any partial sub path that tends at a point along the true optimal path must itself is the optimal path leading up to that point. Therefore the optimal path can be determined by incremental extension of the optimal sub paths. In a Needleman-Wunsch alignment, the optimal path must stretch from beginning to end in both sequences. Dynamic programming methods ensure the optimal global alignment by exploring all possible alignments and finding the best fit between the two sequences. It does this by reading in a scoring matrix that contains values for every possible letter or *nucleotide* match. Needleman-Wunsch finds an alignment with the maximum possible score where the score of an alignment is equal to the sum of the matches taken from the scoring matrix. A scoring matrix system is used to add points to the score for each *match* and subtract them for each *mismatch* and *gaps*, which it gives scores for aligned characters based on a similarity matrix. This algorithm performs alignments with a time complexity of O(*lenA*lenB*) and a space complexity of O(*lenA*lenB*).

The Needleman-Wunsch algorithm consists of three steps : first, initialization of the dot plot, score and the

*traceback* matrices, second, calculation of scores and filling in the score and *traceback* matrices, third, deducing the alignment from the *traceback* matrix. When the steps were implemented there are three matrices produced, the dot plot matrix, the score matrix and the *traceback* matrix.

Local sequence alignment is the method of finding the best possible alignment between any two subsequences of the given sequences. Local alignments are more useful for dissimilar sequences suspected to contain regions of similarity or similar sequence motifs within the larger sequence context. Local alignments identify aligned regions within a larger, often divergent set of sequences. By contrast with global alignment, local alignments are often preferable but can be more difficult to calculate because not all local alignment methods guaranteed optimal solution. However, local alignment aligns DNA more accurately.

Smith-Waterman algorithm uses Dynamic Programming to find the best local alignment between any two given sequences. Based on certain criterion, usually a scoring matrix, scores and weights are assigned to each character to character comparison: positive for exact *matches/substitutions*, and usually negative for *insertion/deletions*. The exact scores are based on a scoring matrix. The scores are added together and the highest scoring alignment is reported. The Smith-Waterman algorithm able to align DNA sequence alignment more accurately without having to align the ends of related DNA which may be highly different. The Smith-Waterman algorithm is implemented by changing only in the filling scores the matrix and traceback in the Needleman-Wunsch algorithm. To further describe the level of similarity between two real Bioinformatics sequences, an affine gap model was introduced to the Smith-Waterman algorithm by O. Gotoh in 1982 [11]. In the affine gap model, the gap is used for the insertion or deletion, to make the alignment more expecting in sequencing.

### 1.2 Optimal Alignment in Sequence Alignment

An optimal alignment, of course, is one that shows the most significant similarities, and the least differences. In order to compare two sequences of characters, a *scoring* system is needed that calculate scores for *match*, *mismatch* and *gaps*. Sequence alignment problem is to produce a pairing of characters from one sequence with the second sequence so that the total score is *optimal*. In pairing characters, *gaps* can be inserted at any position in the sequences. However the order of characters in each sequence must be maintained. An *optimal alignment* of sequence alignment is the one that obtain maximum number of *matches* and minimum number of *mismatches* and *gaps* between the two sequences.

The scoring scheme consists of letter substitution scores, the score for each possible letter alignment and has penalties for *gaps* and *mismatches*. The alignment score is the sum of substitution scores, *gap* and *mismatches* penalties. The alignment score thus reflects goodness of alignment.

Dynamic Programming methods ensure the optimal global alignment by exploring all possible alignments and choose the best [7]. It does this by reading in a scoring matrix that contains values for every possible residue or *nucleotide* match. Optimal global alignment methods allow the best overall score for the comparison of the two sequences to be obtained, including a consideration of *gaps* [9]. Optimal local alignment algorithms seek to identify the best local similarities between two sequences but, unlike segment methods, include explicit consideration of gaps [10].

Optimal alignment is affected by scoring parameters (*gaps*, *mismatch* and *match*), as given a specific scoring system. When the values of the parameters in the scoring system (*gaps*, *mismatch* and *match*) are changed, then it also changed the result of optimal alignment. The optimal alignment of two DNA sequences is an alignment with a score which defines the quality of the match, also mapping one sequence onto the other, possibly with *gaps* [8].

## 2.0 METHODOLOGY

This work is designed to be effective by reducing the *gaps* and length of sequences alignment without sacrificing the result of accuracy, efficient in time and space complexity, and practical for sequence alignments of large genomic regions. To facilitate the implementation based on the specifications for each algorithm used, thus the algorithm design is divided into two divisions, *Needle* (Needleman-Wunsch algorithm) and *Smith* (Smith-Waterman algorithm). The programs implemented by using C Language programming.

### 2.1 Needle Program

*Needle* finds an alignment with the maximum possible score where the score of an alignment is equal to the sum of the matches taken from the scoring matrix. It develops optimal global alignments by computing the similarity between two sequences, *seqA* and *seqB* according to the lengths, *lenA(m)* and *lenB(n)*, respectively, using a Dynamic Programming approach. Though Needleman-Wunsch algorithm is a very effective and thorough means of performing a global alignment, but it is a very slow algorithm, and it demands a great deal of computing time. The algorithm design of *Needle* program is divided into three modules: Dot Plot Matrix , Scoring Matrix and Traceback / Alignment. The followings are the basic terminologies [7] and pointed out according to the work requirements.

#### i. Dot Plot Matrix

For this work, the basic condition for initialization is not used. It initialize the *NeedleArray[i][j]* matrix with a Dot Plot matrix. Dot Plots are an effective visual representation similarities between the two DNA sequences compared. Completed dot plot matrix is

using one point *(1)* for match and zero points *(0)* for mismatch. Dot Plots Matrix is to find repeats letter within sequence by comparing the sequence to itself. Repeats appear as a set of diagonal runs stacked vertically or horizontally. Some implementations vary the size or intensity of the dot depending on the degree of similarity of the two sequences.

It is an excellent approach for finding sequence transpositions, for example, transfers of segments of sequence to a new position on the same or another sequence. This technique is better than the original initialization technique as it helps the calculation of the scoring matrix and indirectly facilitates the search for similarity between the two sequences.

### ii. Scoring Matrix

A scoring function, *NeedleArray*, must exist such that different scores can be assigned to different alignments of two DNA relative to the number of *gaps* and number of *matches* in the alignment. For *seqA* and *seqB* (length *lenA* and *lenB*) the size of scoring matrix is needed for their alignment, where grid dimensions must be *(lenA+1) x (lenB+1)*. In this scoring function, there must be a constant value for *gap*, *match* and *mismatch*, so, let *match* be the score for two letters matching, *mismatch* is the penalty for mismatches, and *gap* is the penalty for inserting a gap.

Scoring function assigned the penalties are constant in value, *match* = 1, *mismatches* and *gap* = 0. Originally, the first row and the first column represent alignments of one sequence with spaces, and *NeedleArray[0][0]* represents the alignment of two empty strings, and is set to zero. For this work, in cell *NeedleArray[0][0]* is not set to zero, but it is continued with the scoring calculation and all other entries are computed with the following formula (refer with : Eq. 1) :

$$NeedleArray[i][j] = max \{ NeedleArray[i-1][j-1] + sub(seqA[i], seqB[j]), NeedleArray[i-1][j] + del(seqA[i]), NeedleArray[i][j-1] + ins(seqB[j]) \}$$

**Equation 1:** Needleman-Wunsch algorithm

Originally, to align and match the two sequences is from left to right but for this work it is start in the bottom row moving up, where *i >= 0* and *j = lenB* until the process of identifying the cell position is ended. This technique is to simplify in calculating the score by referred to the dot plot matrix. While to start in the last column moving left, where *j >= 0*, it begin to test the cell position is the bottom corner of the matrix or the value of *NeedleArray[i][j] = NeedleArray[i][j] + Max* will become the maximum of the above scoring formula. Scoring proceeds from right to left and from the bottom of the matrix to the top until cell (*i = 1*, *j = 1*) is reached. At every position *NeedleArray[i][j]* in the matrix the maximum score found in the row and column beginning at position *NeedleArray[i+1][j+1]* is added to the score of *NeedleArray[i][j]*.

Figure 1 shows complete steps of rescored *NeedleArray[i][j]* matrix. The largest value found in the first row or column indicates the optimal alignment score and represents the number of exact letter matches that the optimal global alignment will produce. The cell containing the highest score also represents the first matching pair of sequence elements in the global alignment. To construct the optimal global alignment, it is begin with the highest *scoring* cell in the column or row beginning one position down and to the right. This step is repeated until it is reaches the far right or bottom of the table, *NeedleArray[i][j]* matrix.

With a simple scoring algorithm such as one that is used here, there are likely to be multiple maximal alignments. Since this is an exponential problem, most Dynamic Programming algorithms will only print out a single solution.

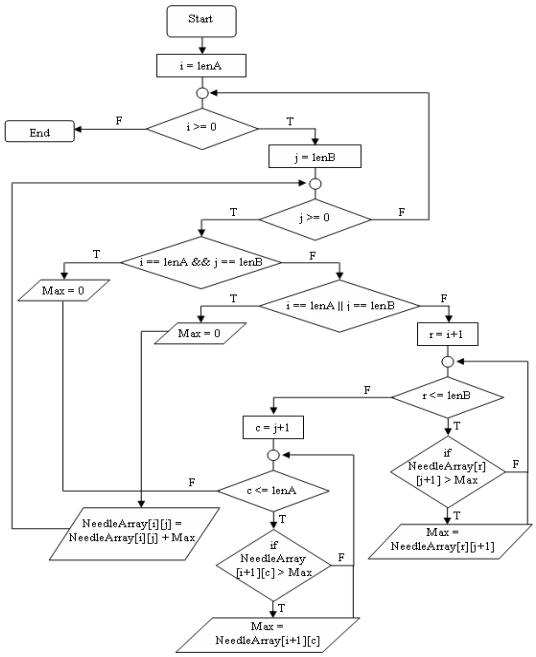



**Figure 1** Flow chart of identifying the cell position and scanning sub row and column to return the maximum score in the matrix, *NeedleArray[i][j]*

### iii. Traceback of *Needle* Program

To find the optimal alignment, this work perform a "*traceback*" by starting at upper-left cell of *NeedleArray[i][j]*, where *i = lenA* and *j = lenB*, at the highest score. When *seqA[i] = seqB[j]* then move to cell *NeedleArray [i-1][ j-1]*, otherwise, move to the larger score of *NeedleArray [i-1][ j]* and *NeedleArray [i][ j-1]*, and mark every cell have come across. To find the final alignment, draw a path from each of the marked letters in which *seqA[i] = seqB[j]* to the next letter, where *seqA[i] = seqB[j]* and has a row and column strictly less than the first letter. The traceback

or alignment process determines the actual alignments that result in the maximum score.

To find the highest scoring position in *NeedleArray[i][j]* matrix, is by scanning the next sub row or column. Then, to extract optimal global alignment from the *NeedleArray* matrix, it have to initialize the position of the cell, where *PositionA = -1* and *PositionB = -1*. It will do four processes to trace the optimal alignment :

a) Find the highest scoring position.
b) Trace matches in the matrix.
c) Insert gaps in *AlignmentB*.
d) Insert gaps in *AlignmentA*.

The process of inserting the gap in *AlignmentB* is executed as shown in Figure 2. Before traceback process move to the left cell of the matrix, *NeedleArray[i][j-1]*, either it skips a row or not, it needs to identify the alignment by determine *needpos[0] > PositionA + 1*, if the condition is T, spaces are assigned to *seqB*.

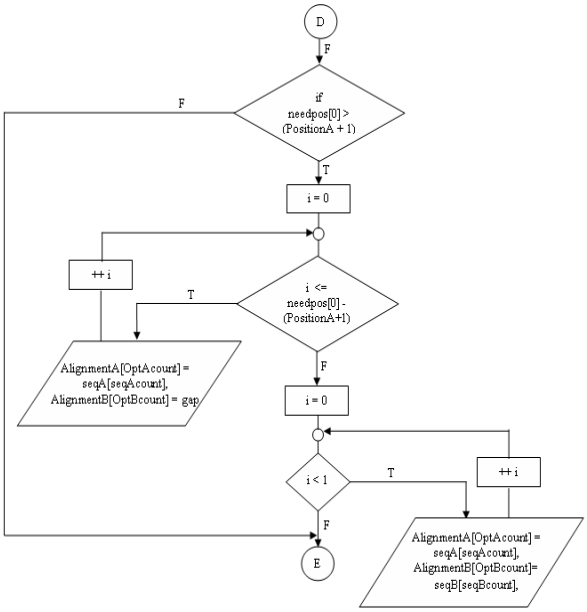



**Figure 2** Flow chart of inserting gap in *AlignmentB*

Figure 3 shows the process of inserting gap in *AlignmentA*. Before traceback process move to up cell of the matrix, *NeedleArray[i-1][j]*, either it skips a column or not, it needs to identify the alignment by determine *needpos[0] > PositionB + 1*, if the condition is T, spaces are assigned to *seqA*.

From the traceback process, diagonal position should be identified to ensure that there is similarity in the alignment and also there is a possibility that mismatch is identified in the alignment. But, there is gap in the alignment when the next alignment is not a diagonal, so traceback has to identify gap in the alignments, either in *seqA* or *seqB*.

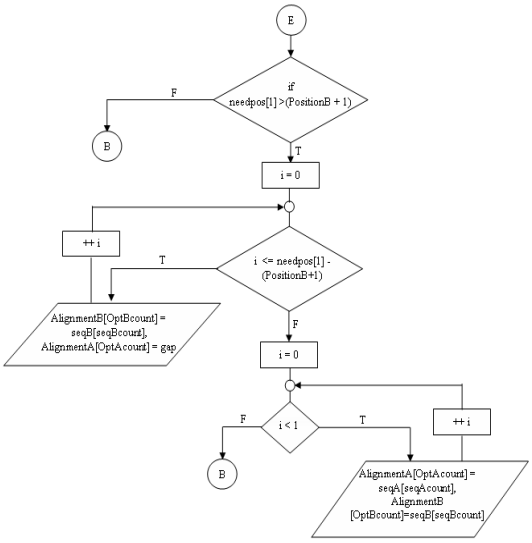



**Figure 3** Flow chart of inserting gap in *AlignmentA*

## 2.2 *Smith* Program

There are three ways that the Smith-Waterman algorithm differs from the Needleman-Wunsch algorithm. First, in the initialization stage, the first row and first column are all filled in with 0. Second, when fill in the table, if a score becomes negative, it put in 0 instead, and add the cell back only for cells that have positive scores. Finally, although Smith-Waterman algorithm is similar to Needleman-Wunsch algorithm but differs primarily in the fact that the traceback occurs from the maximum value in the *lenA* x *lenB* matrix to the first *0* encountered, rather than from the lower right corner to the upper left corner.

*Smith* program is divided into three modules: Initialization, Scoring Matrix and Traceback / Alignment. The formulas to calculate the similarity matrix based on the Smith-Waterman algorithm is shown as below [8] and pointed out according to the work requirements.

### i. Initialization

Scoring function filled the matrix with *0* in the first row, *SmithArray[i][0]*, then, the first column, *SmithArray[0][i]* of the matrix initially filled with *0*. It is to facilitate the traceback process, where it start with the cell that has the highest score and work back until reached a cell with a score of 0. Besides, the edges of the matrix are initialized to *0* instead of increasing gap penalties.

### ii. Scoring Matrix

After the initialization, a matrix fill step is carried out, which fills out all entries in the matrix are computed with the following formula (refer with : Eq. 2):

$$SmithArray[i][j] = max \{ 0, SmithArray[i-1][j-1] + sub(seqA[i], seqB[j]), SmithArray[i-1][j] + del(seqA[i]), SmithArray[i][j-1] + ins(seqB[j]) \}$$

**Equation 2** Smith-Waterman algorithm

In the next phase of the algorithm, the fill phase, also called the induction phase, each one of the cells of the matrix is filled with scores. The scores are then calculated starting in the upper left corner and proceeding outward, the scores of cells *SmithArray[i-1][j]*, *SmithArray*[i-1][j-1], and *SmithArray* [i][j-1].

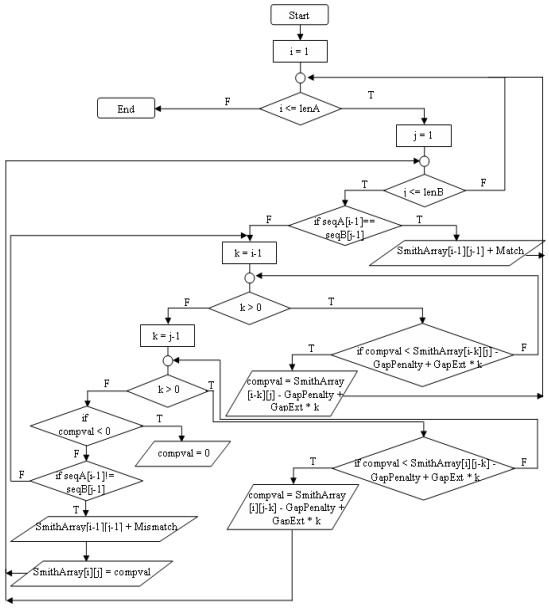



**Figure 4** Flow chart to fill scores in *SmithArray[i][j]* matrix

As shown in Figure 4 to fill the scores in *SmithArray[i][j]* matrix, it needs to define the scoring scheme for *Match = 1*, *Mismatch = -1*, *GapPenalty = 1* and *GapExt = 1*. To read all values in *seqA* and *seqB* it has to be determined by *i <= lenA* and *j <= lenB* so if the condition is T (true) it will identify the match and mismatch of the *seqA* and *seqB*. The largest value found in the first row or column indicates the optimal score of the scoring matrix and represents the number of exact letter matches that the optimal global alignment will produce.

In scoring matrix, only a thing was changed in the Needleman-Wunsch algorithm to obtain the Smith-Waterman algorithm. When filling score in the matrix, it do not let any of the matrix values become negative, and thus consider 0 as potentially being the maximum value of the three other cases (where *seqA[i] = seqB[j]*, or there is a gap in *seqA* or a gap in *seqB*) and add the pointer back only for cells that have positive scores. By not letting any of the values go below zero, stop considering regions of high dissimilarity which have no good alignments. This allows the algorithm to focus on only those regions of the DNA which are similar. It also keeps track of which cell has the higher score and then it need that for the traceback.

### iii. Traceback

To find the optimal alignment it started the traceback with finding the highest score (opimal score) in sub columns and rows cell in the matrix and it occurs by following the path through the maximum scores back until a 0 value is reached. So for each cell, not only must a scoring value be held, but a directional value must be kept as well indicating how it got from one cell to the next in the optimal path.

When the sequences become aligned their similarity scores align along a diagonal, if there was a gap, the values line up vertically or horizontally. Thus, in tracing matches in the matrix, the cell position is moved diagonal from current position, it simply align and it means matches has founded. the position is not diagonal, so it inserting gap into *seqB*. For inserting gap in *seqA*, it has to compare for ali elements of *seqB*. Once the seqA and seqB is compared and it generated match, mismatch and gap between the sequences, then it work the optimal sequence is aligned.

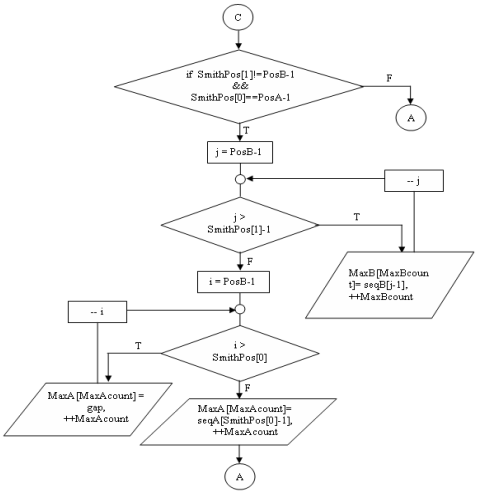



**Figure 5** Flow chart of trace match in the matrix and inserting gap in *MaxB*

Figure 5 is the Section *B* process for tracing matches in the matrix and the highest score in sub columns and rows, then it align strings to the *SmithPos[]*. When the condition of *(SmithPos[0] == PosA-1) && (SmithPos[1] == PosB-1)* is T, so the cell position is moved diagonal from current position, it simply align and increment the counters, where it means matches has founded. Then, it initialize *MaxA[MaxAcount] = seqA[Smithpos[0]-1]* and *MaxB[MaxBcount] = seqB[Smithpos[1]-1]*.

Otherwise, if the condition is not true and the position is not diagonal, so it inserting gap in *MaxB* and it set dashes "-" to *seqB*.

Figure 6 is the Section C process for inserting gap in *MaxA* and the condition of *SmithPos[0] == PosA-1) && (SmithPos[1] != PosB-1)* must be T. It also has to compare for all elements of *seqB* between *PosB* and *SmithPos[1]*. Then, it set dashes "-" to *seqA* and inserting gap in *MaxA*. Once the seqA and seqB is compared and it generated match, mismatch and gap between the sequences.

The Smith-Waterman algorithm contains no negative scores in the path matrix it creates. The algorithm starts the alignment at the highest path matrix score and works backwards until a cell contains zero.

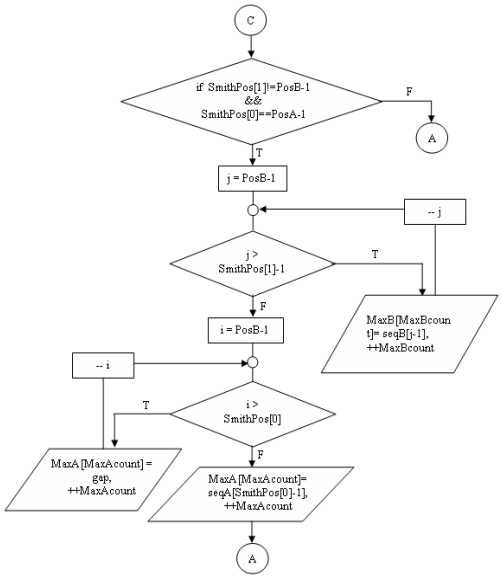



**Figure 6** Flow chart of inserting gap in *MaxA*

### 2.3 Optimal Alignment Measurement

In order to verify the accuracy and consistency of measurements obtained in Needleman-Wunsch (global) and Smith-Waterman (local) algorithm the data is compared with the establish system from EMBOSS for global and local sequence alignment. A complete data collection of the sequence alignment is analyzed according to the parameters of measurement that consists of *similarity*, *gap*, *mismatch* and *execution time*. Results of *similarity* percentage are found from the matches in the sequences aligned and showed that more matches aligned, where higher percentage of *similarity* calculated. While, results of *gap* and *mismatch* percentages are found from the gaps and mismatches in the sequences aligned. Although, *gap* and *mismatch* are very important in similarity searching but its use should be reduced in order to achieve the effectiveness of the space complexity.

Table 1 shows the summary of optimal sequence alignment in determining the effectiveness of the accuracy, space and time complexity. To achieve a good performance in sequence alignment, it is important to reduce the *gap*, *mismatch* and *execution time*, but in searching similarity, it should be increased.

The results of the analysis is compared with EMBOSS. When the result in similarity is lower than EMBOSS, its mean that EMBOSS is reliable in finding optimal sequence alignment, instead of results in gaps and mismatch. When the result in similarity lower than EMBOSS, the results in gaps and mismatch are higher than EMBOSS, and the result in execution time is lower than EMBOSS its mean that EMBOSS is better than the other applications (*Needle* and *Smith*) in finding the optimal sequence alignment and have a good performance in time and space complexity, and vice versa.

**Table 1** Summary of optimal sequence alignment

| Parameter | Mean | Std. Dev. | Corr. |
|---|---|---|---|
| Similarity | High | High | High |
| Gap | Low | Low | Low |
| Mismatch | Low | Low | Low |
| Execution Time | Low | Low | Low |

## 3.0 RESULTS AND DISCUSSION

Experiments for Needleman-Wunsch and Smith-Waterman algorithms are conducted on 600 different pairs of sequences from the Genbank sequence database, National Center for Biotechnology Information (NCBI) [13]. Data in this experiment are made up with DNA sequences from *Gorilla gorilla* (Western Gorilla) as *seqA* and *Pan Troglodytes verus* (West African Chimpanzee) as *seqB*, where *seqA* is a query sequence and *seqB* is a compare sequence. There are 600 different data from chimpanzees that are run paired with a data from gorilla. During the sequence alignment process only a DNA sequence of Western Gorilla is used and it is held constant, as *seqA* while *seqB* are made up from 600 data which differ for DNA sequences of West African Chimpanzee. These data could be used to represent a comparison of evolution paths between related species, between Western Gorilla and West African Chimpanzee.

### 3.1 Analysis on Similarity

The similarity of two DNA sequences can be defined as the best score among all possible alignments between them. Sequence alignment is based on the identification of similar sequences and similarity is used to specify relationships between the two DNA sequences, as it is the preferred choice for biological applications. Results of similarity percentage are

found from the matches in the sequences aligned and the summary of the measurement data for the similarity is shown in Table 2. More matches aligned from the two DNA sequences therefore higher percentage of similarity is computed. Results of similarity are collected from the implementation of the 600 data and it is executed for each method. The highest percentage of mean for similarity is *Needle* with 50.66% and the establish findings system, *EMBOSS (local)* has the lowest mean percentage which is 45.34%. According to the result, it is do not necessarily that *Needle* is the most accurate method compare to others. Although the mean similarity of *Needle* is in a high range but the result of the correlation test must be taken into account as well.

**Table 2** Summary of measurement data of similarity for the four methods

| Algorithm(s) | Mean (s) | Std. Dev. (s) | Corr. (s) |
|---|---|---|---|
| Needle | 50.66 | 1.72 | 0.3115 |
| EMBOSS (global) | 43.85 | 1.76 | |
| Smith | 47.54 | 1.6 | 0.008 |
| EMBOSS (local) | 45.34 | 1.78 | |

While, the results of standard deviation for the four methods are not much different and it is still at a low range, where it shows that from the implementation of the 600 data there is a variation in terms of the result of standard deviation. It is illustrating that there are differences in similarity of the DNA sequences aligned. This means that the standard deviation is in a low range but still in positive area and it is showing that the results coupled closely with mean and the variance distribution is not wide.

Data verification and consistency of the results of *Needle* and *Smith* were compared direct with the establish system, *EMBOSS (global)* and *EMBOSS (local)*. Although *Needle* has the highest mean similarity but the correlation test with *EMBOSS (global)* is not significant, where the data generated by the *Needle* is a good correlation only with his own. This shows that *Smith* is better than the three methods because the correlation test between *Smith* and *Emboss (local)* is significant at 0.008. In the DNA sequence alignment, there is no perfect similarity and that is why the smallest correlation test means it will lead to perfect similarity. However, the correlation between the two distributions is in positive range and showing that both distributions refer to the same mean and variance, where it is close to the actual establish finding systems.

### 3.2 Analysis on Gap

In an alignment, each character of a DNA sequence can be aligned to another sequence in one of two ways, the DNA character can be aligned with any character in the other sequence, or the DNA character can be aligned with a gap inserted into the other sequence. To achieve a good performance in sequence alignment, it has to reduce the gaps and increasing the matches for the sequences aligned. The implication of this is the length of the aligned sequences will be reduced too and the achievement in space and time complexity will be acquired. Results of gap percentage are found from the gaps of the sequences aligned. The summary of the measurement data for gap is shown in Table 3.

**Table 3** Summary of measurement data of gap for the four methods

| Algorithm(s) | Mean (s) | Std. Dev. (s) | Corr. (s) |
|---|---|---|---|
| Needle | 38.69 | 1.66 | 0.0799 |
| EMBOSS (global) | 41.67 | 2.44 | |
| Smith | 34.06 | 1.61 | 0.0827 |
| EMBOSS (local) | 42.05 | 2.6 | |

*Smith* has the lowest gap percentage which is 34.06%. In sequence alignment, when reducing the gaps is acquired automatically it is reducing the length of the aligned DNA sequence and it is also able reduce the execution time. This is means the probability to obtain a good accuracy in sequence alignment was high. As referring to the results of analysis on similarity in Table 2, Needle has the highest percentage of similarity compared to others, while in gap percentage it has 38.69% but it is still in a good range of accuracy for sequence alignment.

For the results of standard deviation for gap, the difference range of standard deviation for Needle and Smith is quite similar which are 1.66% and 1.61%. While, *EMBOSS (global)* and *EMBOSS (local)* give high value of standard deviation compare to this project methods. But, the results of standard deviation for the four methods are still at a low range. It is shows that from the implementation of the 600 data there is a variation in terms of the result of standard deviation. It is illustrating that there are differences in gaps of the DNA sequences aligned. This is means that the standard deviation is in a low range but still positive and it is showing that the results coupled closely with mean and the variance distribution is not wide.

The comparison of data verification and consistency of the result for the four methods show that the results coupled closely with mean and the variance distribution is not wide. While, the results for the correlation between the two distributions is in a positive range and showing that both distributions refer to the same mean and variance, where it is also close to the actual establish finding systems.

Although *Needle* has the lowest mean for gaps and the correlation test with *EMBOSS (global)* is quite

significant, 0.0799, but it is not is do not necessarily that *Needle* is the good performance method compare to others. However, according to the analysis on similarity and gaps, it shows that *Smith* is better than the other three methods because the correlation test between *Smith* and *Emboss (local)* is significant at 0.0827, where *Smith* is in a good range for similarity and gap.

### 3.3 Analysis on Mismatch

Results of mismatch percentage are found from the mismatches of the sequences aligned. The mismatch is occurs when the similarity in sequences aligned is increases and the gap is decreases. Either increased or decreased of mismatch are predicated to wax and wane of percentage for similarity and gap. This is means, mismatches usually occurs between the matches and gaps in obtaining a good alignment. It indicated by a blank meaning that the DNA is not identical, but they have some of the same chemical or structural properties.

**Table 4** Summary of measurement data of mismatch for the four methods

| Algorithm(s) | Mean (s) | Std. Dev. (s) | Corr. (s) |
|---|---|---|---|
| Needle | 10.65 | 1.09 | 0.1501 |
| EMBOSS (global) | 14.48 | 1.89 | |
| Smith | 12.27 | 1.1 | 0.1499 |
| EMBOSS (local) | 12.75 | 2.04 | |

Results of mismatch percentage are found from the mismatches in the sequences aligned, as shown in Table 4, the establish findings system, *EMBOSS (global)* is the highest mean for mismatch with 14.48%. The lowest mean percentage is *Needle* with 10.65%. But, it is not necessarily that *Needle* is the most accurate method compare to others, because, although the mean of mismatch for *Needle* is in a low range but the result of the correlation test, analysis on similarity and gap must be taken into account as well. While, *Smith* and *EMBOSS (local)* have nearly similar value with 12.27% and 12.75%.

The results of standard deviation for mismatch, the difference range of standard deviation for *Needle* and *Smith* is quite similar which are 1.09% and 1.1%, where they have lower percentage of standard deviation *than EMBOSS (local)* and *EMBOSS (global)*. But, the results of standard deviation for the four methods are still at a low range. It is shows that from the implementation of the 600 data there is a variation in terms of the result of standard deviation. It is illustrating that there are differences in mismatches of the DNA sequences aligned. This is means that the standard deviation is in a low range but still positive and it is showing that the results coupled closely with mean and the variance distribution is not wide.

The data verification and consistency of the correlation test of mismatch between *Needle* and *EMBOSS (global)* is quite significant which is 0.1501. According to the analysis on similarity, gaps and mismatches, it shows that *Smith* is better in performance than the other three methods because the correlation test of mismatch between *Smith* and *Emboss (local)* is significant at 0.1499, where *Smith* is in a good range for similarity, gaps and mismatch. However, the correlation between the two distributions is positive and showing that both distributions refer to the same mean and variance, where it is close to the actual establish finding systems.

## 4.0 CONCLUSION

This work is to provide a method that is capable to solve problem in the search of similarity accuracy for the comparison in sequence alignment. The similarity accuracy is measured when a method can increase the *similarity* and reducing the *gap* and *mismatch*, and it occurs from the result of the process for searching, comparing and aligning of the two DNA sequences. The process will not occur if the two DNA sequences are not related. The main purpose of the experimental is to evaluate the performance of Needleman-Wunsch and Smith-Waterman algorithms and it is compared with EMBOSS (global) and EMBOSS (local), the establish system for sequence analysis that has been used quite widely. It is compared in order to verify the accuracy and consistency of the measurements.

In the DNA sequence alignment, there is no perfect similarity and that is why the smallest correlation test means it will lead to perfect similarity. To achieve a good performance in sequence alignment, it is important to reduce *gap*, *mismatch* and execution time, and increasing the matches for the sequences aligned. When the gaps have been reduced, then the length of the aligned sequences will be reduced too. Thus, the achievement in space and time complexity will be acquired. Based on the analysis carried out on *similarity*, *gap*, *mismatch* and execution time, it showed that *Smith* is better than *Needle*, *Emboss (local)* and *Emboss (global)*, because all the results of the analysis showed that the data generated by *Smith* is significant, where it meets the requirements in reducing gap and mismatch, and increasing similarity.

## References

[1] S. K. Moore. 2000. *Understanding the Human Genome*. IEEE Press. 11: 34-35.
[2] Todd, J. Vision and Aoife McLysaght. 2003. Computational Tools and Resources in Plant Genome Informatics. In P. Christou & H. Klee (Ed.). *Handbook of Plant Biotechnology* John Wiley & Sons. 1552.
[3] K. Charter, J. Schaeffer, and D. Szafron. 2000. Sequence Alignment using FastLSA. *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*. 239-245.

[4] S. F. Altschul, T. L. Madden, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. 1997. Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. *Nucleic Acids Research*. 25: 3389-3402.
[5] D.J. Lipman. and W. R. Pearson. 1985. Rapid and Sensitive Protein Similarity Searches. *American Association for the Advancement of Science*. 227(4693): 1435-1441.
[6] Z. Michalewicz, D. B. Fogel. 2004. *How to Solve It: Modern Heuristics*. New York: Springer.
[7] S. B. Needleman and Christian, D. Wunsch. 1970. A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Sequences. *Journal of Molecular Biology*. 48: 443-453.
[8] Temple, F. Smith and Michael, S. Waterman. 1981. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*. 147: 195-197.
[9] A. J. Ropelewski, H. B. Nicholas and D. W. Jr. Deerfield. 2002. Strategies for Searching Sequence Databases. *Journal of Biotechniques*. 28: 1174-1178.
[10] W. R. Taylor, T. P. Flores and C. A. Orengo. 1994. Multiple Protein Structure Alignment. 10.
[11] A. Davidson. 2004. A Fast Pruning Algorithm for Optimal Sequence Alignment. *Proc. 2nd IEEE International Symp. Bioinformatics and Bioengineering*. 49-56.
[12] B. Gärtner and J. Matoušek. 2006. Understanding and Using Linear Programming: Elementary Introduction for Mathematicians and Computer Scientists, Berlin. Springer. 1858-1870. Information on http://www.proteinscience.org.
[13] NCBI. Genomic Biology. 2015. Information on http://www.ncbi.nlm.nih.gov/GenBank.