Proceedings of
ICISIP - 2005

Improved Algorithms for DNA Sequence Alignment and Revision of Scoring Matrix

SUBHRA SUNDAR BANDYOPADHYAY, SOMNATH PAUL and AMIT KONAR
E-mail: subhra_ju@yahoo.co.in, appusom@yahoo.com, amit_konar@vsnl.net
Electronics and Telecommunication Department
Jadavpur University, Kolkata, India

ABTRACT
Sequence alignment is the procedure of comparing two (pair-wise alignment) or more (multiple sequence alignment) sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences. Classical methods like Needleman-Wunsch (for global alignment) and Smith-Waterman (for local alignment) suffer from the drawback that it involves a large number of computational steps and has to statically allocate a large section of memory for computer implementation. Moreover the PAM matrix used as a scoring scheme for sequence comparison is unsuitable when we are considering only a few of the total 20 amino acids. In this paper, we propose alternate methods to obtain global and local alignment between two sequences using direct comparison and compare the performance of our algorithms with that of the classical method. We also propose an alternate scoring scheme based on fuzzy concept and discuss its advantage over conventional PAM matrix.

1. INTRODUCTION

The dynamic programming algorithm for local alignment between two DNA sequences proposed by Smith and Waterman is a very well known and versatile algorithm, and has widely been referred in the domain of Bio-informatics. The scoring matrix construction and trace-back causes a significant degradation in the runtime of the above algorithm. A simple analysis of the algorithm reveals that it has a significantly high time-complexity lying between $n \times m$ and $n \times m^2$, and space complexity of the order of $n \times m$, where n and m denote the lengths of the shorter and longer sequences respectively.

The classical method to obtain global alignment is the Needleman-Wunsch method. However this method suffers from the drawback that it involves a large number of computational steps and has to statically allocate a large section of memory for computer implementation.

With the advent of fast and reliable technology for sequencing nucleic acids and proteins, centralized databases were created to store the large quantity of sequence data produced by labs all over the world. This created a need for efficient programs to be used in queries of these databases.

The dynamic programming method for computing similarities and optimal alignments between two sequences makes them unsuitable for searching large databases. To speed the search, novel and faster methods have been developed. In general, these methods are based on heuristics and it is hard to establish their theoretical time and space complexity. Nevertheless, the programs based on them have become very important tools and these techniques deserve very careful study.

This paper covers the following topics:
In section 2 we discuss the local algorithm for obtaining the optimal alignment between two sequences. In section 3 we discuss the global algorithm for obtaining the optimal alignment between two sequences. In section 4 we discuss fuzzy scoring matrix. Section 5 is the conclusion, which discusses some important features of the proposed algorithms.

SECTION 2: LOCAL ALIGNMENT

SECTION 2.1: SMITH-WATERMAN ALGORITHM

A modification of the dynamic programming algorithm for sequence alignment provides a local sequence alignment giving the highest scoring local match between two sequences. The rules for calculating the scoring matrix values are:

i) The scoring system must include negative scores for mismatching,
ii) When a dynamic programming scoring matrix value becomes negative, that value is set to zero, which has the effect of terminating any alignment up to that point.

The alignments are produced by starting at the highest scoring position in the scoring matrix and following a trace-back path from that position up to the position that scores zero. For two sequences $a=a_1 a_2 - - - a_n$ and $b=b_1 b_2 - - - b_n$, where $H_{ij}=H(a_1a_2 - - a_i, b_1 b_2 - - b_j)$, then,

$$H_{ij}=\max\{H_{i-1,j-1}+S(a_ib_j),\ \max_{x\geq 1}(H_{i-x,j}-w_x), \max_{y\geq 1}(H_{i,j-y}-w_y), 0\}.$$

Where, H_{ij} is the score at the position i in sequence a and position j in sequence b. $S(a_ib_j)$ is the score for aligning the characters at positions i and j, w_x is the penalty for a gap of length x in sequence a and w_y is the penalty for a gap of length y in sequence b.

SECTION 2.2: WEAKNESS AND SCOPE OF IMPROVEMENT

Smith-Waterman algorithm requires much larger number of computational steps, since we have to form the matrix and trace-back accordingly. This algorithm also suffers from much larger space complexity due to storage of matrix. We now discuss a new algorithm, which requires much less amount of time and space complexity.

SECTION 2.3: PROPOSED ALGORITHM

The proposed algorithm consists of six main steps. Let s_1 and s_2 be two sequences that we need to align.

Step 1: Consider the first element of s_2 to be c, we compare c with each element of the sequence s_1.

Step 2: If search is successful i.e. match is found, then we assign c as the first element of s_4 and the corresponding match i.e. say pth element of s_1 as first element of s_3. (s_3 and s_4 are the two output strings.)

Step3: Then we perform the following steps:

```
IF the second element of $s_2$ matches with (p+1)th element of $s_1$.
        Assign the second element of $s_2$ as the second
        element of $s_4$ and (p+1)th element of $s_1$ as the
        second element of $s_3$.
END IF
ELSE
        IF the third element of $s_2$ equals (p+2)th element of $s_1$.
                Assign the second and the third
                elements of $s_2$ as the second and the
                third elements of $s_4$ respectively and
                (p+1)th , (p+2)th element of $s_1$ as the
                second and the third element of $s_3$.
        END IF
        ELSE IF second element of $s_2$ equals (p+2)th element of $s_1$.
                Store (p+1)th , (p+2)th element
                of $s_1$ as the second, third element
                of $s_3$ and a gap , second
                element of $s_2$ as the second, third
                element of $s_4$.
        END ELSE IF
        ELSE IF third element of $s_2$ equals (p+1)th element of $s_1$.
                Assign second, third element of
                $s_2$ as the second, third element
                of $s_4$ and a gap, (p+1)th element
                of $s_1$ as the second, third
                element of $s_3$.
        END ELSE IF

        ELSE
                Stop comparison.
        END ELSE
END ELSE
```

(In this algorithm, we consider that, there is maximum one gap permissible for each searched element.)

Step4: The score of the alignment is found by using standard scoring matrix (PAM 250, BLOSUM 62) and for gap penalty, a score of –4 is introduced.

Step5: Step1 to step4 is repeated for all other elements of s2 and if matching is found more than once, then step1 to step4 is repeated for each match.
Step6: The alignment corresponding to the maximum score is chosen ---- which is the optimum local alignment.

SECTION 3: GLOBAL ALIGNMENT

SECTION 3.1: NEEDLEMAN-WUNSCH ALGORITHM

Needleman-Wunsch used dynamic programming in order to obtain global alignment between two sequences. Global alignment, as the name suggests takes into account all the elements of the two sequences while aligning the two sequences .We can also call it as an "end to end " alignment.
In Needleman-Wunsch algorithm, a scoring matrix of size m*n (m being the length of longer sequence and n being that of the shorter sequence)is first formed. The optimal score
At each matrix position is calculated by adding the current match score to previously scored positions and subtracting gap penalties. Each matrix position may have a positive, negative or 0 value.
For two sequences:
$a=a_1a_2......a_m$
$b=b_1b_2......b_n$
where $S_{ij}=S(a_1a_2.......a_m, b_1b_2......b_n)$ then
The element at the i, jth position of the matrix $S_{i,j}$ is given by

$$S_{i,j}=\text{Max}\{\ S_{i-1,j-1}+s,\quad \underset{x=>1}{\text{Max}}\ (S_{i-x,j}-w_x),\quad \underset{y=>1}{\text{Max}}\ (S_{i,j-y}-w_y)\ \} \qquad (1)$$

where S_{ij} is the score at position i in the sequence and j in the sequence b, $S(a_ib_j)$ is the score for aligning the characters at positions i and j,w_x is the penalty for a gap of length x in the sequence a and w_y is the penalty for a gap of length y in the sequence b. After the S matrix is filled up, to determine an optimal alignment of the sequences from scoring matrix, a method called trace back is used. The trace back keeps track of the position in the scoring matrix that contributed to the highest overall score found. The positions may align or may be next to a gap, depending on the information in the trace back matrix. There may exist multiple maximal alignments.

SECTION 3.2: WEAKNESS AND SCOPE FOR IMPROVEMENT

A study of the Needleman-Wunsch algorithm reveals two shortcomings of the algorithm:-
For increasing length of the sequences in comparison, the computational complexity becomes quite large and the size of the scoring matrix also might be huge. The implementation of the above algorithm on a computer using static allocation might be difficult. It is not guaranteed that the results given by the Needleman_Wunsch algorithm implementation will give global alignment. This means if there is a requirement of an "end to end " matching, the results might not comply with the requirement. Hence a new approach may be taken up in aligning the two sequences by direct comparison method taking into consideration that computational steps have to be minimized and "end to end" matching is fulfilled.

SECTION 3.3: PROPOSED ALGORITHM

If the length of the two sequences in comparison differs by a considerable margin then we resort to method1, if the lengths are comparable (equal or smaller by a value of 1 or 2), then we resort to method2.
Let us denote the longer sequence by S_1 and the shorter (or of same length) sequence by S_2 .Let the optimally aligned sequences obtained in method1 and method2 be denoted by S_3 and S_4.Let the length of the sequence S_1 be **m** and that of S_2 be **n**.
The stepwise explanations of the two methods are given below:

SECTION 3.3.1: METHOD1

Step1) We consider the **ith** element of the shorter sequence S_2 and compare it with **(m-n+1)** elements of S_1, starting from **(x+1)** to **(x+1+m-n+1)**, where x denotes the position of the last positioned element of S_3.We start with **i=1.**
Step2) A counter to count the number of gaps previously introduced in S_3 is taken and assigned **0** for the first comparison. If the number of gaps previously

introduced is less than **(m-n)**, then we compare the **ith** element to the elements of the range mentioned above. If during comparison, a match is found between the elements of S_2 and an element of S_1 lying between **(x+1)** to **(x+1+m-n+1)**, we introduce gaps in S_3 from **(x+1)** to the position just before where the match was found. The matched element is then placed at the position of match.

Step3) If the number of gaps introduced exceeds **m-n**, then we place the elements of S_2 in S_3 from position **x+1** onwards.

Step4) If during comparison from **x+1** to **(x+1+m-n+1)** elements, no match is found, then we place the element of S_2 that is in comparison at the **(x+1)** th position in S_3.

Step5) The above procedure is followed with S_1 and S_2 reversed and S_4 is obtained. Score of S_3 and S_4 are compared and the best is chosen. If the score of S_4 is more, then reversed S_4 is the optimal alignment.

SECTION 3.3.3: METHOD2

Step1) Let us consider the **ith** element of S_1 and **jth** element of S_2 .If S_{1i} matches with S_{2j}, then we place S_{1i} and S_2j at the **(x+1)** th positions of the optimal alignment pair S_3 and S_4which are to be obtained, where **x** is the position of the last positioned element of S_3 and S_4.We start with **i=0,j=0** and **x=-1**(considering the first element of the sequence to be **0**).

Step2) If S_{1i} does not match with S_{2j}, then we check whether S_{2j} matches with S_{1i+1} or S_{1i+2}.If a match is found with S_{1i+1},then we place a gap under S_{1i} in S_4 and place S_{2j} under S_{1i+1}.For match with S_{1i+2},two gaps are to be introduced. If a match is found for both S_{1i} and S_{2j}, then the higher score is taken.

Step3) If the above matching do not occur, then we check whether S_{1i} matches with S_{2j+1}or S_{2j+2}.Suitable number of gaps are introduced in the same manner as in step 2,only difference being the gaps are now in sequence S_3.

Step4) If all the above comparisons fail, then we place S_{1i} and S_{2j}one above the other in S_3 and S_4 respectively.

Step5) The same procedure is continued until we come to the end of one of the sequences (S_1 or S_2). End gaps are introduced if the length of S_3 is greater than the length of S_4 or vice versa.

Step6) S_1 and S_2 may be reversed and aligning procedures are followed. The score for forward and backward comparison are compared and the best is taken.

SECTION 4: FUZZY SCORING MATRIX

The conventional PAM matrix is static in nature. By this we mean that even when we are working with a few selected amino acids we have to consider the effect of other amino acids also because of the manner in which the PAM matrix values have been calculated (as discussed above). However we can model a new scoring matrix based upon fuzzy concept, which is dynamic in nature i.e. the matrix values depend upon the number of amino acids that are being considered.
Instead of taking the logarithm of the odds score, we are considering only the odds score. We are considering odds score of each amino acid and normalize these values i.e. we are evaluating the ratio of two odds score.

The significance of the ratio of two odds score is as follows:
The numerator expresses the expected number of cases where amino acid changes to another amino acid due to mutation, which signifies that the particular genetic site concerned is unstable i.e. prone to mutation. The denominator expresses the expected number of cases where amino acid remains unchanged, which signifies the stability of the region or the site. Their ratio as a result means the hyper-mutable tendency of a specific genetic site. Further the significance of the ratio, of course, would also depend on its usage or application. If the structure of the protein is the question, for example, then the hyper mutability is an enough reason for further deductions. However for any functional usage, a phenomenon called "codon degeneracy" should also be taken into account.
We represent the normalized odds scores of each amino acid as the membership values of that amino acid and construct the membership graph.

Now to construct the scoring matrix, we are considering only four amino acids namely C, T, A, G.

The PAM-250 matrix for these amino acids is given by:

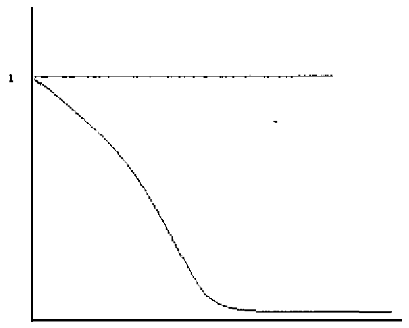
	C	T	A	G
C	12			
T	-2	3		
A	-2	1	2	
G	-3	0	1	5

The odds scores of amino acid C is given by:

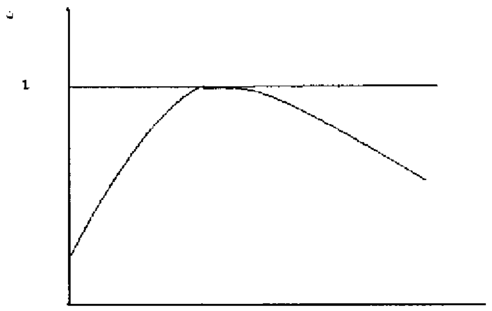
	C	T	A	G
Odds scores	15.85	0.63	0.63	0.50
Normalized odds score	1.00	0.04	0.04	0.03

The odds scores of amino acid T is given by:

	C	T	A	G
Odds scores	0.63	1.995	1.259	1.00
Normalized odds score	0.316	1.00	0.63	0.50



MEMBERSHIP GRAPH OF AMINO ACID C



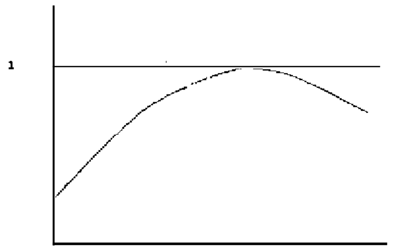
MEMBERSHIP GRAPH OF AMINO ACID T

The odds scores of amino acid A is given by:

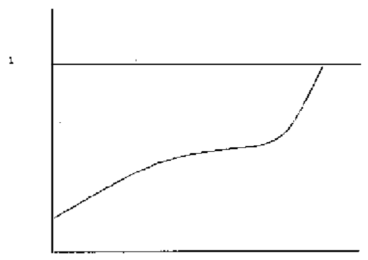
	C	T	A	G
Odds scores	0.63	1.259	1.585	1.259
Normalized odds score	0.397	0.794	1.00	0.794

The odds scores of amino acid G is given by:

	C	T	A	G
Odds scores	0.50	1.00	1.259	3.16
Normalized odds score	0.158	0.316	0.398	1.00



MEMBERSHIP GRAPH OF AMINO ACID A



MEMBERSHIP GRAPH OF AMINO ACID G

The scoring matrix considering only these four amino acids is given below:

	C	T	A	G
C	1.00	0.04	0.04	0.03
T	0.316	1.00	0.63	0.50
A	0.397	0.794	1.00	0.794
G	0.158	0.316	0.398	1.00

By considering more number of amino acids, the fuzzy scoring matrix can be modified accordingly.

SECTION 5: CONCLUSION

As evident from computational analysis the proposed algorithms require less number of comparison steps than the exiting methods and also does not involve the formation of two dimensional scoring and trace back matrices.

For the proposed algorithm of global alignment the following two points should be noted:

1) For method1 the best score may be missed if the length of the shorter sequence is too close to the length of the longer string or is otherwise very short. For two sequences of comparable length meth●d2 may be used, if on the other hand the difference is too large then it is better to use local alignment.
2) For method2 the introduction of end gaps may be undesirable, this may be avoided if a check is kept on the difference of the gaps inserted in the aligned sequences and this difference is not allowed to a pre-decided value.

In fuzzy scoring matrix, we have considered directed mutations, so $f_{ab} \neq f_{ba}$, where f_{ab} denotes the number of times the mutation a↔b was observed to occur. Also we have also considered all the scores of the fuzzy scoring matrix to be positive. The negative score for mismatching in PAM matrix is replaced by small positive score in fuzzy scoring matrix. The fuzzy scores are dynamic, since the scores depend on the total number of amino acid considered. But the PAM scores are static.

REFERENCES

[1] SMITH-WATERMAN ALGORITHM --- Local Alignment Algorithm Using Dynamic Programming.
[2] NEEDLEMAN-WUNSCH ALGORITHM---Global Alignment Algorithm Using Dynamic Programming.
[3] GOTOH O. ---An improved algorithm for matching biological sequences.
[4] HUANG X., MILLER W. ---A time-efficient, linear-space local similarity algorithm.
[5] CHAO K.M., HARDISON R.C., MILLER W. --- Recent developments in linear-space alignment methods
[6] JOAO SETUBAL AND JOA● MEIDANIS. --- Introduction to computational molecular biology, University of Campinas, Brazil, December 1997.
[7] WARREN J. EWENS AND GREGORY R. GRANT. --- Statistical methods in bioinformatics: an introduction, Springer-Verlag New York, 2001.