A Modified Dynamic Parallel Algorithm for Sequence Alignment in BioSequences

S. Nirmala Devi
Research Scholar
Bharath University
Chennai – 600 073. India

S.P. Rajagopalan, PhD.
Professor Emeritus, Dr. M.G.R Educational and Research Institution University
Chennai-600095. India

ABSTRACT
This paper presents a Modified Dynamic Parallel Algorithm for Sequence Alignment in BioSequences. DNA sequence alignment between two or more bio sequences using algorithms is a complex problem due to its applicability in the field of biology. This algorithm achieves its efficiency in using computational resources by making a M X N matrix , M represents the length of first sequence and N represents the length of the second sequence .This modified Dynamic algorithm performs calculation to fill the three main diagonal cells without evaluating other cells in the matrix.

The proposed model is based on Index Based Pattern Matching using Multithreading [13] to obtain the optimal alignment using Multithreading. The executed results indicate that with the proposed algorithm Memory Efficiency and Fast Execution are achieved over the well-known dynamic programming approach Needleman-Wunsch and Hirschberg approach.

Keywords
DNA operations, sequence alignment, Multithreading, score.

1. INTRODUCTION
The study of algorithms for different character strings is one important area of algorithm design. Among the most important is the efficiently searching for substrings or generally different patterns in different databases. In many instances we do not want to find a subsequence exactly, but something similar. The process of discovery of patterns proves to be essential for biological researches and business scenarios.

The basic unit of an organism is DNA and the basic units of DNA are nucleotides and each nucleotide is one of the following four types: adenine (A), guanine (G), cytosine (C) and thymine (T). It can be viewed as a long sequences of A's, G's, C's and T's. It is very difficult to retrieve necessary information from the sequence when the size of the database grows. Its length ranges from a few hundred to several billions of nucleotides of different species. Thus, it is a complicated task to find out the degree of similarity and the degree of difference between nucleotide (DNA/RNA) and amino acid (protein) sequences.

The knowledge of a DNA sequence and gene analysis can be used in several biological, medicinal and agriculture research fields such as: possible disease and abnormality diagnoses, forensics, pattern matching, biotechnology etc.,

Alignment program tries to find the best alignment between two sequences given the scoring system. There are two approaches to make the alignment algorithm.

1.1 Local Alignment
An alignment that searches for segments of the two sequences that match well. There is no attempt to force entire sequences into an alignment, just those parts that appear to have good similarity, according to some criterion [3].

Example:

```
LGPSTKDFGKISESREFDN
          || ||
LNQLERSFGKINMRLEDA
```

1.2 Global Alignment
An alignment that assumes that the two sequences are basically similar over the entire length of one another. The alignment attempts to match them to each other from end to end, even though parts of the alignment are not very convincing[3].Using the same sequences as above, we would get

```
LGPSTKDFGKISESREFDN
|         | | | |   |
LNQLERSFGKINMRLEDA
```

To find the similarity between biological sequences several software tools were built. The most commonly used web tool is Basic Local Alignment Search Tool (BLAST) [4].

Another tool that is use for multiple sequence alignment is DIALIGN which combines both local and global alignment features and used dynamic programming in its algorithm [5].

2. BACKGROUND AND RELATED WORK
Several methods for sequence alignment have been proposed and they have their own advantages and limitations [1] [7][8][9].

In [1] Amine Dhraief presented a Parallel method for computing Longest Common Subsequences.

In [7], Yinan Li presented a method called WHAM, which employs novel hash-based indexing and bitwise operations for pair wise alignment.

In [8] Isokawa et al. presented a method to deal with multiple sequence alignment using a Genetic Algorithm.

Most commonly used algorithm for local sequence alignment is Smith-Waterman Algorithm [9].

The proposed algorithm is based on Dynamic Programming Needleman_wunsch Algorithm and Hirschberg method.

2.1 Needleman_Wunsch Algorithm

Given two input strings x and y, this method build a matrix F such that the entry F[i, j] is the score of the optimal alignment of x[1..i] and y[1..j] , where an alignment is defined to be the path from the top-left corner to the bottom-right corner of the Needleman-Wunsch Matrix[10][11].

This method calculates value for each cell for the entire matrix.

The path is composed of the three moves:

Diagonal if xi aligns with yj either via a match or mismatch

Right if xi aligns to a gap

Left if yj aligns to a gap.

The steps of Needleman-Wunsch Algorithm is as follows:

1. Initialization:

F(0,0)=0

F(i,0)=-1

F(0,j)=-1

2. Main Iteration

For each i = 1 to M

For each j = 1 to N

F(i,j) =max

{ F(i − 1, j − 1) + s(xi, yj), case 1

F(i − 1, j) − d , case2

F(i, j − 1) − d , case3 }

ptr(i, j) =

{ DIAG , if case 1

LEFT , if case 2

UP , if case 3 }

3. Termination

Performance:

Time : O(NXM) to fill out the entire matrix

Space : O(NXM) to store all the trace back pointers.

This method fills all the cells of the matrix which requires more computation time and more memory space.

2.2 Hirschberg Algorithm

Hirschberg presented algorithms for multiple sequence alignment by using divide-and conquer techniques. The divide and conquer techniques can effectively reduce the space complexity for multiple sequence alignment. First, the cutting points of sequences are found and the sequences are separated into two sets of subsequences, according to the cutting points. Then the dynamic programming techniques are applied for aligning the subsequences in the subsets. [2][3].The alignment based on divide-and conquer method has the benefit for increasing the speed compared to sequence alignment using dynamic programming method.

3. MODIFIED DYNAMIC PARALLEL ALGORITHM USING MULTITHREADING

In this MDPA method the very large size DNA sequences are divided into subsequences depending upon the sizes of sequences which have the maximum length. The main idea by using multithreading is to solve the sequence alignment problem on a single CPU machine is to make the alignment simultaneously in a timesharing manner [12][13].

The subsequences are stored in separate arrays and the modified dynamic parallel algorithm is executed on these sequences.

Example:

Consider the Input sequences A and B as shown in Fig.1

A=	G	C	G	C	A	T	G	G	A	T	T	G	A	G	C	G	A
B=	T	G	C	G	C	C	A	T	T	G	A	T	G	A	C	C	

Fig 1: Input Sequences A and B

Here the length of sequence A=17 and the length of sequence B = 16.

Based on the Length of the sequence having maximum length, here length of A=17 and the above sequences are split into subsequences as shown in Fig.2 and they are passed as parameter to various threads.

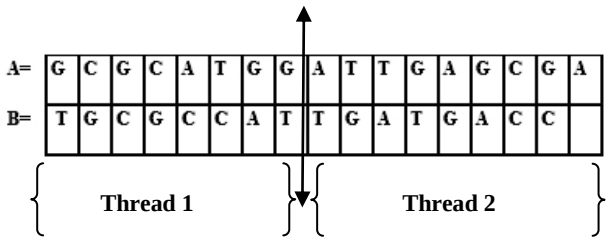

Fig 2: Splitting of sequences A and B

If then length of the sequences are very long we can specify the cutting point and further it can also be subdivided in number of subsequences and various threads can also be created based on the number of subsequences.

The algorithm steps are as follows:

Input: Two Sequences A of m characters and B of n characters

Given two sequences A, B, create a matrix D of m x n (m is the length of A and n is the length of B).

Output: The score value and an alignment of the two sequences in which all characters in both sequences participate.

Step 1: [initialization of variables]

match = +1

mismatch = - 1

gap = -1

```
D[0][0] =0
D[1][0]=-1
D[0][i]=-1
```

Step 2: fill the value in the three diagonal cells (below main diagonal, main diagonal, and above main diagonal)

```
For each i =1 to m
For each j = 1 to n
If (i==j)
D[i][j]=D[i-1][j-1]+match;
Dir(i,j)= Diagonal
Else if (i!=j)
if(i-j<=1 && i-j>=-1) {
D[i][j]= max{
D[i-1,j1]+mismatch, D[i-1,j]+gap ,D[i,j-1]+gap}
if max= D[i-1,j-1]+mismatch
dir(i,j) = diagonal
If max= D[i-1,j]+ gap
dir(i,j)=Left
If max= D[i,j-1]+gap
Dir(i,j) = Up
```

Step 3: Trace the Backtracking Matrix

```
i=A.length
j=B.length
score=D[i][j];
 while (i > 0 && j > 0) {
     if ((D[i][j] == mD[i-1][j-1] )+ match)) {
          newSeqA += A[i-1];
          newSeqB += B[j-1];
          i--;
          j--;
          continue;
     } else if (mD[i][j] == mD[i][j-1] - 1) {
          newSeqA += "-";
          newSeqB += B[j-1];
          j--;
          continue;
     } else {
          newSeqA += A[i-1];
          newSeqB += "-";
          i--;
          continue;
     }
}
```

Step 4: print score value and the alignment of two sequences.

To validate the proposed method let us take two sequences A=”AGTA” and B=”ATA’.

Create a matrix of size m x n, m is the length of sequence A and n is the length of sequence B.

The execution of the MDPSA method is as follows:

Step 1: Initializing values for match gap and mismatch.

D[0][0]=0

D[0][1]=-1

D[1][0]=-1

Dir-specifies the track back for optimal alignment.

Step 2: Fill values only in the cells of three main diagonals

For i=1 to 4

For j=1 to 3

If (i==j)

D(1,1)=D(0,0)+isMatch(A(i-1),B(j-1))

D(0,0)=0+1=1

Dir=diagonal

If(i!=j)

D(1,2)=max(D(0,1),D(1,1),D(0,2)+isMatch(A(i-1),B(j-1))

Max(-1,1,N value)= 1

Dir= left - The maximum value

And for all the three diagonal values, the value of the matrix is as follows (Fig.3):

j= / i=	0	1 A	2 T	3 A
0	0	-1	0	0
1 A	-1	1	0	0
2 G	0	0	0	-1
3 T	0	0	1	-1
4 A	0	0	0	2

Fig 3: Filled Matrix of the new Algorithm MDPSA

D(4,3) represents the optimum score and the value is 2. Trace back the optimal alignment from this optimal score position by the dir value to get the optimal alignment.

4. EXPERIMENTAL RESULTS

The proposed method is implemented using Java jdk1.6 on a Intel i3 PC for dealing with the DNA sequence alignment problem. To make comparison between the proposed method and NeedlemanWunsch and Hirschberg method, sequences A=”AGTA” and B=”ATA” are given as Input to find the optimal solution for the given sequences.

Fig 4. shows the Scoring Matrix and optimal alignment for the given Sequences using existing NeedlemanWunsch method.

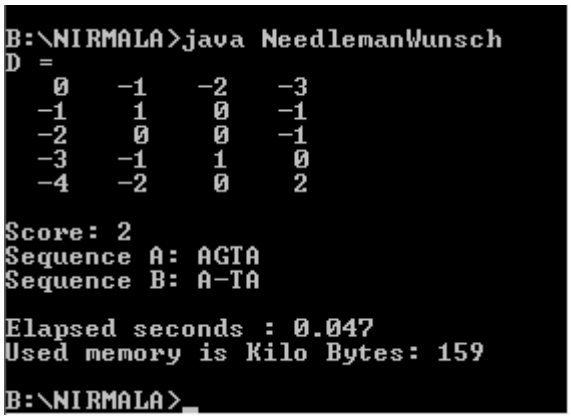

Fig 4: the output alignment and score matrix for the sequences A=ATA and B=AGTA by using Needleman-Wunsch algorithm.

The last cell (5, 4) has the maximum score Alignment value.

Fig. 5 shows the result of the proposed method Modified Dynamic Parallel algorithm before using Multithreading.

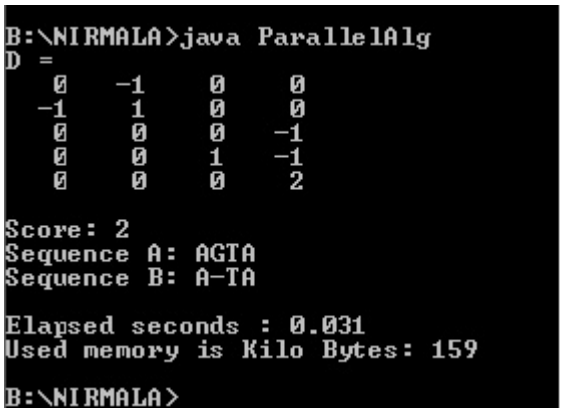

Fig 5: the output alignment and score matrix for the sequences A=ATA and B=AGTA by using the new Algorithm MDPSA

To validate the proposed method using Multithreading, sequences A=”AGTAACATA” and B=”ATAACTA” are given as Input and Fig.3 shows the optimal alignment and scoring matrix.

Sequence A is divided into two subsequences as a1=”AGTA” and a2=”ACATA”

Sequence B is divided into two subsequences as b1=”ATA” and b2=”ACTA”

Optimal score matrix and optimal Alignment is for a1, b1 and a2, b2. The result is shown in Fig.6.

From these results, MDPSA algorithm finds the same optimal solution as Needleman-Wunsch algorithm and Hirschberg algorithm and it calculates only the three diagonal values (main diagonal, below main diagonal and above main diagonal) and this algorithm will not manipulate values for other cells, thus reducing execution time.

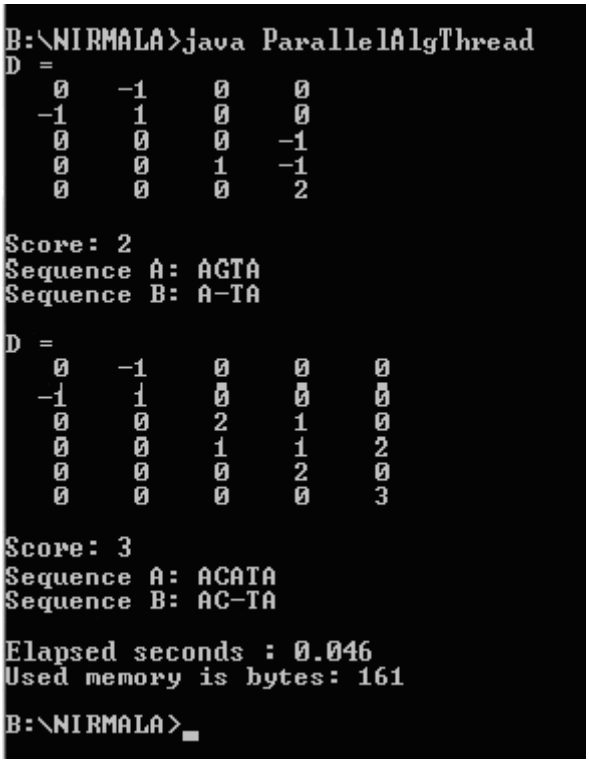

Fig 6: the output alignment and score matrix of the new Algorithm MDPSA using Multithreading.

Table 1. shows the comparison between the specified existing algorithms with the proposed technique for the Sequence Set X as A=”ACTA”, B=”ATA” and Y as A=”ACAAGACAGCGT”, B= “AGAACAAGGCGT”.

Table 1. Comparisons of existing Algorithms with MDPSA

Algorithms	Sequence Set	Memory Used in KB	CPU Time in Seconds
FDPSA	X	159	0.031
	Y	160	0.093
FDPSA(Thread)	X	161	0.046
	Y	161	0.078
Needleman-Wunsch	X	159	0.047
	Y	160	0.109
Hirschberg	X	154	0.063
	Y	148	0.103

From the comparisons the new approach is efficiently improving the time complexity as it is based on multithreading technique. The use of sequence analysis is very broad and Modified Dynamic Parallel algorithm can improve system performance. This Multithreaded implementation improves the CPU utilization and increases the time efficiency.

5. CONCLUSION

This paper describes the implementations of finding the Sequence Alignment based on bioinformatics algorithm in a highly parallel way and it is very easy to implement. This method can also be used for sequence alignment in protein sequences and also for business and marketing research to analyze series of purchase over time. The proposed method

showed significant improvement in terms of the efficiency in finding the solution.

6. REFERENCES

[1] Amine Dhraief, Parallel Computing the Longest Common Subsequences (LCS) on GPUs: Efficiency and Language Suitability, INFOCOMP 2011 : The first International Conference on Advanced Communications and Computation.

[2] Hirschberg, D.S 1977. Algorithms for the longest common subsequence problem. Journal of ACM ,24:664-675.

[3] A.Aho, D.Hirschberg and B. Jullman –Bounds on the complexity of the longest Common Subsequence Problem, J. Assoc. Comput.Mach., Vol.23, No.1, 1976.

[4] Biology, Cambridge University Press, New York, 1997. BioInformatics Educational Resources Documentation (online), European Bioinformatics Institute United Kingdom.Available: http://www.ebi.ac.uk/2can/tutorials/protein/align.html

[5] BLAST, http://blast.ncbi.nlm.nih.gov/Blast.cgi,

[6] DIALIGN, http://dialign.gobics.de

[7] Yinan Li , WHAM : A High Throughput Sequence Alignment Method , SIGMOID '11, June 12-16, 2011.

[8] Isokawa,M. Wayamaa,M., and Shimizu T.1996 . Multiple sequence alignment using a Genetic Algorithm. Proceedings of the Seventh Workshop on Genome Informatics, 7:176-177.

[9] Smith, T. F. and M. S. Waterman, Identification of common molecular subsequences, Journal of Molecular Biology, 147:195-197, 1981.

[10] Needleman S, Wunsch.,"A general method applicable to the search for similarities in the amino acid sequences of two proteins", J Mol Biol. 1970, 48:443-453.

[11] Rong X, Jan 2003, Pairwise Alignment - CS262 - Lecture 1 Notes(online), Stanford University. Available: http://ai.stanford.edu/~serafim/cs262/Spring2003/Notes/1.pdf

[12] Deitel P. and Deitel H., Java How to Program, Prentice Hall, 2003.

[13] S.Nirmala Devi , "An Index based Pattern Matching using Multithreading ", International Journal of Computer Applications (0975 – 8887) Volume 50- No.6, July 2012.