IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. I (Nov. - Dec. 2016), PP 16-21
www.iosrjournals.org

A Study to the different implementation approaches for the Grid YM-Algorithm DNA alignment

Mohamed Assal, Ahmed Said, Soha Souliman and Ahmed Essam
Faculty of Computers and Information MTI University Cairo, Egypt

Abstract: *DNA Multiple sequence alignment is common bioinformatics application that determines the similarity between a new sequence with other exist sequences. Needleman-Wunsch algorithm is the most famous algorithm for DNA global alignment. Unfortunately, it is based on sequential computing so it has a problem of being slow. Recently purposed algorithm (YM-Algorithm) aim to overcome this sequential computing limitation. This paper presents a different implementation approaches for the YM algorithm in a real environment. Finally, the paper study the experimental results of applying these implementations approached in a realistic parallel environment.*

Keywords: *DNA Computing, Computational biology, Distributed computing, Grid computing, Bioinformatics*

I. Introduction

Pairwise Sequence Alignment is a sequence alignment of two biology sequences such as DNA, RNA, or protein. The result of the task can be used to infer sequence homology and conduct phylogenetic analysis to assess the sequences shared evolutionary origins. The accuracy and execution time are major factors requiring the attention of researchers. Some popular alignment approaches used are exact solution,progressive methods, iterative methods, or methods based onHidden Markov Models. Each method has its advantages anddisadvantages. Biologists are the persons who decided suitablemethod to process their biological data[1] [2].

The sequence alignment is the problem withexponential complexity. Over the years, researcher efforts in finding different algorithms or mathematical models thatrequire low computational cost as well as ensure accuracy[1]. Multiple sequence alignment is computationally intensive problem and classified as a NP-Hard problem [3] [4]. Sequences can be aligned across their entire length (global alignment) or only in certain regions (local alignment). Local sequence alignment plays a major role in the analysis of DNA and protein sequences [5] [6].

The two general models view alignments in different ways:the first considers similarity across the full extent of thesequences (a global alignment); the second focuses onregions of similarity in parts of the sequences only (a localalignment). It is important to understand these distinctions, toappreciate that sequences are not uniformly similar, and thereis no value in performing a global similarity on sequencesthat have only local similarity. Therefore, finding local similaritymay produce more biological meaning and sensitive resultthan finding optimal alignment over entire length of thesequence[7][8].

A lot of popular algorithms were proposed, researches in the field of bioinformatics has grown significantly in the recent years as demands for more computing power increased. The solutions to these demands usually involve using parallel and/or distributed techniques. Grid Computing is an evolving technology to provide high performance computing in a virtual environment composed of a large number of computers connected through network[9].

Recently a new alignment algorithm has been purposed in [10]. The algorithm aimed to overcome the time consumption problem of dynamic programming methodology used by the Needleman-Wunsch algorithm. Unlike Needleman-Wunsch algorithm, which is sequential, this algorithm aims to take advantage of the computational power of the grid computing in order to improves the execution time required for the alignment process [10]. Through this paper, this algorithm will be called YM-Algorithm.

This paper provides a comparative study to the different implementation approaches for the YM alignment algorithm in order to run it on the Grid Developing System (GDS) [11]. This study is significant as the YM-Algorithm author simulate their algorithm using the MATLAB and never test it on a realistic grid environment.

II. Global Alignment

The data of sequence are partitioned into DNA sequences and protein sequences. Each DNA sequence consists of four types of base A, T, C and G and each protein sequence is made up of 20 types of amino acid, hence any sequence can be represented as a string over specific alphabet.

DNA sequence alignment is a representation of the similarity between two or moresections of genetic code. It is used to compare these sections in a quantitative way.Biologists use the comparisons to discover evolutionary divergence, the origins ofdisease, and ways to apply genetic codes from one organism into another[12].

A. Pair wise Sequence Alignment

Pair wise sequence alignments are used to find diagnosticpatterns that characterize the two DNA families; to detect ordemonstrate homology between new sequences and existingfamilies of sequences[7].

B. Global Sequence Alignment

The idea of aligning two sequences (of possibly different sizes) is to write one on top of the other, and break them into smaller pieces by inserting spaces in one or the other so that identical Subsequences are eventually aligned in a one-to-one correspondence - naturally, spaces are not inserted in both sequences at the same position. In the end, the sequences end up with the same size. The following example illustrates aglobal alignment between the sequences A="ACAAGACAGCGT" and B="AGAACAAGGCGT"[7].

```
A = A C A A G A C A G - C G T
    |   | | |   | |   | | |
B = A G A A C A - A G G C G T
```

Fig-1 Global Alignment of two sequences

The objective is to match identical subsequences as far as possible. In the example, nine matches are highlighted with vertical bars. However, if the sequences are not identical, mismatches are likely to occur as different letters are aligned together. Two mismatches can be identified in the example: a "C" of A aligned with a "G" of B, and a "G" of A aligned with a "C" of B. The insertion of spaces produced gaps in the sequences. They were important to allow a good alignment between the last three characters of both sequences [13].

Once the alignment is produced, a score can be assigned to each pair of aligned letters, called aligned pair, according to a chosen scoring scheme. We usually reward matches and penalize mismatches and gaps.
The similarity of two sequences can be defined as the best score among all possible alignments between them. Note that it depends on the choice of scoring scheme. In the next sections, the problem of finding the best alignment of two sequences (an alignment that gives the highest score) will be addressed[13].

C. Substitution Matrices

In the previous example, fixed scores were given for matches, mismatches and gap penalties. However, biologists frequently use scoring schemes that take into account physicochemical properties or evolutionary knowledge of the sequences being aligned. This is common when protein sequences are compared.

For instance, for some reason one might want to penalize the mismatch of an aspartic acid (D) with leucine (L) more heavily than a mismatch between the same aspartic acid with, say, histidine (H). [13].

This type of scoring schemes is called *alphabet-weight* scoring schemes, and is usually implemented by a substitution matrix. Currently, two types of amino acid substitution matrices are being largely used by biologists for practical protein sequence alignment: PAM and BLOSUM. They were developed from different concepts but have the same structure. In fact, they are a series of matrices with varied degrees of sensibility.

The PAM matrices (acronym for point accepted mutations) are extrapolated from data obtained from very similar sequences to reflect an amount of evolution producing on average one mutation per hundred amino acids. The BLOSUM matrices (acronym for blocks substitution matrix), in contrast, were developed to detect more distant relationships [14]. In particular, BLOSUM50 and BLOSUM62 are being widely used for pairwise alignment and database searching [15].

III. YM Algorithm

In [10], The Authors presented a new global alignment algorithm that can be implemented using parallel computing (such as grid computing) to overcome the Needleman-Wunsch sequential computing limitation.

Their algorithm assumes that length of *SEQ1* is greater than or equal length of *SEQ2*. The algorithm searches for the maximum length consecutive match in *SEQ1* and*SEQ2*. The authors defined the consecutive match as the consecutive sequence of residues in *SEQ1* and *SEQ2* such that the cumulative score increases as long as one residue of the first consecutive sequence is compared with its corresponding residue of the second consecutive sequence. After finding the maximum length consecutive match, *SEQ1* and *SEQ2* are split into three sub-sequences each, the left side of consecutive match (*L_SEQ1, L_SEQ2*), the maximum consecutive match (*M_SEQ1, M_SEQ2*), and the right side of consecutive match (*R_SEQ1, R_SEQ2*). Again, search for the maximum consecutive match is applied recursively for the (*L_SEQ1, L_SEQ2*) and (*R_SEQ1, R_SEQ2*). For each

recursive iteration, the algorithm add the gaps required to keep the maximum consecutive match in each part aligned, finally the algorithm concatenate these three parts[10].

The algorithm divided into three main sub-modules: Consecutive match sub-module (CM), split sequence sub-module (SS), and combine sub-sequences submodule (CS)[10].

The CM algorithm obtains all possible consecutive matchesand saves the parameters of each match in a parameters table (*P*_TABLE). These parameters include *WORD1* (the consecutive matched residues of*SEQ1*) start index, *WORD2* (the consecutive matched residues of *SEQ2*) start index, the consecutive match length (*W_LENGTH*), and the consecutive match score (*SCORE*)[10].

Eventually, *P_TABLE* is searched to obtain the maximum consecutive match (*M_SEQ1*, *M_SEQ2*) as well as other maximum consecutive match parameters (*M_SEQ1* start index, *M_SEQ2* start index, W_LENGTH).

Figure 2 shows the pseudo code of CM algorithm[10].

CM algorithm

```
Input: SEQ1, SEQ2, P_TABLE, FIRST_TIME
If FIRST_TIME
{
    FIRST_TIME ← FALSE
    For k=0 to length(SEQ1)
    {
        SCORE ← 0
        For j=0 to length(SEQ2)
        {
            SEQ1_INDEX ← j+k (modulo length(SEQ1))
        If S (SEQ1_INDEX,j) > 0
        {
                If W_LENGTH=0
Get WORD1_START_INDEX,
                                        WORD2_START_INDEX
                SCORE ← SCORE+ S (SEQ1_INDEX, j)
                Add SEQ1(SEQ1_INDEX) to WORD1
                Add SEQ2(j) to WORD2
                WLENGTH ← WLENGTH+1
                Add the match parameters to P_TABLE
        }
        Else
        {
                SCORE ← 0
                WORD1 ← ""
                WORD2 ← ""
                W_LENGTH ← 0
        }
        }
    }
}
Get the maximum consecutive match parameters from match parameters table
Output: P_TABLE, Maximum consecutive match parameters
```

Fig-2 CM Algorithm

The SS algorithm obtains (*L_SEQ1*, *L_SEQ2*)), (*R_SEQ1*, *R_SEQ2*) and the match parameters table (*P_TABLE*) to generate match parameter tables (*LP_TABLE*, *RP_TABLE*) for (*L_SEQ1*, *L_SEQ2*) and (*RSEQ1*, *RSEQ2*), respectively. *LP_TABLE* only considers the consecutive matches in *L_SEQ1 and L_SEQ2*, and *RP_TABLE* only considers the consecutive matches in*R_SEQ1 and R_SEQ2*. Figure 3 shows the pseudo code of SS algorithm[10].

```
CS algorithm
Input: Maximum consecutive match parameters, P_TABLE
    Get L_SEQ1,L_SEQ2
    Get R_SEQ1, R_SEQ2
    For each record in P_TABLE
        If match parameters belong to L_SEQ1, L_SEQ2 {
            If (L_SEQ1 overlaps M_SEQ1) or
       (L_SEQ2 overlaps M_SEQ2)
    Modify parameters
            Add these parameters to LP_TABLE
    }
    else if match parameters belong to LSEQ1, LSEQ2 {
            If (RSEQ1 overlaps MSEQ1) or
       (RSEQ2 overlaps MSEQ2)
    Modify parameters
            Add these parameters to RP_TABLE
}
Output: L_SEQ1, L_SEQ2, R_SEQ1, R_SEQ2, LP_TABLE,
    RP_TABLE
```

Fig 3 CS Algorithm

CS algorithm simply concatenates (*L_SEQ1, M_SEQ1, R_SEQ1*) and concatenates (*L_SEQ2, M_SEQ2, R_SEQ2*).

For each iteration, gaps might be added to either *L_SEQ1*or *L_SEQ2*, to make their length equal and keep maximum length consecutive match aligned. Similarly, gaps might be added to either*R_SEQ1* or *R_SEQ2* for the same reason.

IV. Grid Computing

A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality of-service requirements. At the basic level, a grid can be viewed as an aggregation of multiple machines (each with one or more CPUs) abstracted to behave as one "virtual" machine with multiple CPUs. The GDS is a .NET based computational grid environment implemented in MTI University [11]. The GDS Grid allow the seamless aggregation of the computing power of multiple distributed machines connected through network into a virtual super computer. The GDS grid computing framework was conceived with the aim of making grid construction and development of grid software as easy as possible without sacrificing flexibility, scalability, reliability and extensibility [11].

GDS has practical capabilities of connecting up to 4096 workstations. In addition, GDS is hardware scalable in which workstations could be easily replaced with a high-end server through the GDS plug and play agent feature. The GDS will automatically utilize the new powerful resources in the new connected agents [16].

V. The Implemented Approaches

The authors of YM-Algorithm tested their algorithm in simulated environment using MATLAB. We implemented the YM-Algorithm in Microsoft C# 6 and .NET Framework 4.5.2. The first implementation approach was exactly as the authors of the YM-Algorithm presented in their pseudo code. In which, The CM submodule called for the first run with the flag *FIRST_TIME* is true, while called with the flag *FIRST_TIME* is false for all subsequent iterations. During the implementation of the first approach we notice that the parameter table *P_TABLE* generated in the first time is complete and valid for all iterations. Accordingly, In the second approach the parameter table P_TABLE will be readjusted instead of building it in each subsequent iteration.The start indices and/or the length of each record in the *P_TABLE*will be readjusted according to the selected best consecutive match. Consequently, we decide to implement those two approaches and test them side to side.

VI. Experiments And Discussions

In this paper, it has been decieded to firstly test the YM-Algorithm 2 implmenation approaches in a real parallel environment before put them to test in the GDS grid. The implementedapproaches tested using a workstation with the specifications in TABLE I.The experiments ran for each approach on a single core (sequential), 2 Cores, and 4 Cores. In addition to comparing the sequential results to a standard implementation of unmdified NeedlemanWuanch. The experiments done on a 2 sequences of an equal variable lengths starting from 10,000x10,000 characters up to 40,000x40,000 characters.

TABLE II shows some of the experiment results of the 1st implmentation approach.TABLE III shows some of the experiment results of the 2nd implementation approach.

TABLE II. FIRST APPROACH RESULTS

Sequences Length	Cores		
	Single	Dual	Quad
10000	14.586	10.548	9.020
11000	19.994	14.459	12.364
12000	26.980	19.511	16.684
13000	22.154	16.021	13.700
14000	27.793	20.099	17.187
20000	55.451	40.100	34.291
25000	90.421	65.390	55.916
30000	116.634	84.346	72.127
35000	142.848	103.303	88.337
40000	169.061	122.259	104.547

TABLE III. SECOND APPROACH RESULTS

Sequences Length	Cores		
	Single	Dual	Quad
10000	26.603	20.778	19.879
11000	30.972	24.901	26.257
12000	37.490	31.270	31.163
13000	42.573	36.145	32.720
14000	48.643	40.478	43.732
20000	106.036	92.732	86.700
25000	145.550	126.371	118.577
30000	188.113	163.324	153.178
35000	230.676	200.276	187.778
40000	273.239	237.228	222.378

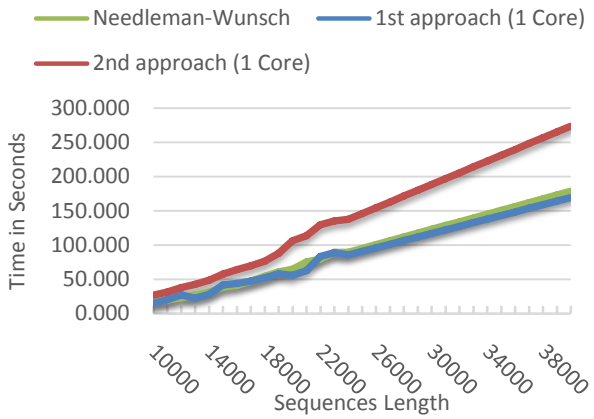

Figure 4 Sequential Processing

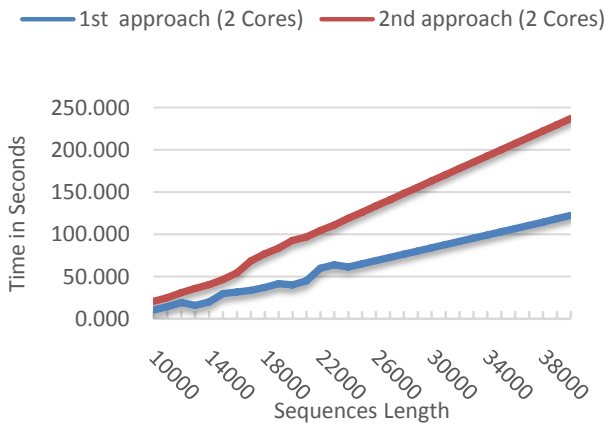

Figure 5 Processing using 2 Cores

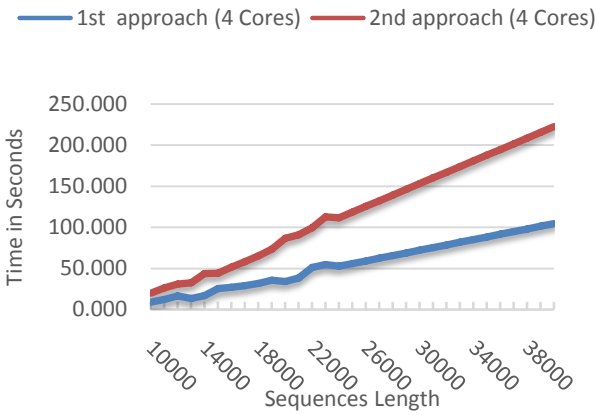

Figure 6 Processing using 4 Cores

The pervious expirements show that the YM-Algorithm has a huge data dependence between threads running in parallel which limits the processing on parallel multi-cores environment. In a real grid environment, there is more overheads to provide access to these shared data for the working agents. These overheads due to many factors such as network bandwidths, … etc. Therefore,its obvious that the authors of the YM-Algorithm assumption to apply the algorithm on grid is not completely suitable due to the need to a shared memory environment.

VII. Conclusion

The paper presented a couple of implementation approaches for the newly proposed YM-Algorithm for DNA multiple sequence alignment. Unfortunately, the experiments done in this work proved that the authors of the paper assumption to apply the algorithm on grid is not completely satisfactory in the shared memory environment. Consequently, the YM-Algorithm will suffer a lot in a real grid computing environment due to the low compuatation to communication ratio. After a series of experiments, Its observable that the plain YM-Algorithm is preferred to be implemented as a sequential rather than parallel implementation. The results show that the 2nd impementation approach for the YM-Algorithm has the nature to be implemented in parallel mode than in sequential mode.

References

[1]. L. V. Vinh, T. V. L. Lang, N. Thi Thu Du and V. H. B. Chau, "Multiple Sequence Alignment on the Grid Computing," International Journal of Computer Science and Telecommunications, vol. 3, no. 7, pp. 46-51, 2012.
[2]. M. Assal, A. Said, D. Mohamed and N. Osama, "A Study to the effect of task granulation for the DNA multiple sequence alignment on Grid Computing," IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS),, vol. 5, no. 3, 2015.
[3]. L. Jiang and W. T., "On the complexity of multiple sequence alignment," Journal of Computational Biology, vol. 1, no. 4, p. 337–348, 1994.
[4]. S. S., L. Y. and Q. Yang, "A polynomial time solvable formulation of multiple sequence alignment," Journal of Computational Biology, vol. 13, no. 2, p. 309–319, 2006.
[5]. S. Altschul, W. Gish, W. Miller, E. W. Myers and D. J. Lipman, "Basic Local Alignment Search Tool," Journal of Molecular Biology, vol. 215, no. 3, pp. 403-410, 1990.
[6]. W. Pearson and D. Lipman, "Improved Tools for Biological Sequence Comparison," in Proceedings of The National Academy of Sciences - PNAS, USA, 1988.
[7]. M. Assal, A. Said, D. Mohamed and N. Osama, "A Study to the effect of task granulation for the DNA multiple sequence alignment on Grid Computing," IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), vol. 5, no. 3, pp. 301-305, 2015.
[8]. S. Vasantharathna, A. Kunthavai and R. Karuppayya, "AGAligner – DNA Local Sequence Alignment Using Alchemi Grid," IRACST – Engineering Science and Technology: An International Journal (ESTIJ), ISSN: 2250-3498, vol. 2, no. 3, 2012.
[9]. I. Foster, "What is the Grid? A Three Point Checklist," [Online]. Available: http://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf. [Accessed 1 July 2016].
[10]. M. Shalaby and Y. Kamal, "Grid Computing-based Sequence Alignment," in Int'l Conf. Bioinformatics and Computational Biology (BIOCOMP'14), 2014.
[11]. A. Said, M. Assal and M. Bakr, "An Enhanced framework for Grid Computing Developing System (EGDS)," Managerial Research Journal, Consultancy Research & Development Center, 2012.
[12]. M. Lehman, "Experiments with Algorithms for DNA Sequence Alignment," Simpson College, Indianola, Iowa.
[13]. S. Junior and d. C. Anibal, "Sequence Alignment Algorithms," King's College London, University of London, London, 2003.
[14]. D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.
[15]. R. Durbin, S. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis, Probabilistic Models of Proteins and Nucleic Acids, Cambridge University Press, 1998.
[16]. A. Said, Design and Building of a Framework for Grid Computing Developing System, Cairo: Arab Academy for Science, Technology & Maritime Transport, December 2012.