33rd Annual International Conference of the IEEE EMBS
Boston, Massachusetts USA, August 30 - September 3, 2011

ABS: Sequence Alignment By Scanning

Talal Bonny and Khaled N. Salama
Electrical Engineering Program
King Abdullah University of Science and Technology (KAUST)
Thuwal, Kingdom of Saudi Arabia
Email: {talal.bonny, khaled.salama}@kaust.edu.sa

Abstract—
Sequence alignment is an essential tool in almost any computational biology research. It processes large database sequences and considered to be high consumers of computation time. Heuristic algorithms are used to get approximate but fast results. We introduce fast alignment algorithm, called 'Alignment By Scanning' (ABS), to provide an approximate alignment of two DNA sequences. We compare our algorithm with the well-known alignment algorithms, the 'FASTA' (which is heuristic) and the 'Needleman-Wunsch' (which is optimal). The proposed algorithm achieves up to 76% enhancement in alignment score when it is compared with the FASTA Algorithm. The evaluations are conducted using different lengths of DNA sequences.

I. Introduction

In the field of Bioinformatics, the amount of generated data is growing exponentially. For example, the genomic sequence data at Genbank is doubling every 15 months. As the growing of these data over the past several years has far exceeded the growth in processor performance, researchers find new algorithms/techniques to speedup the process and to improve the performance. These algorithms [3] [4] which are based on the dynamic programming method such as the Needleman-Wunsch [6] (for global alignment) and Smith-Waterman algorithms [2] (for local alignment), provide optimal alignment in a time that is proportional to the product of the lengths of the two sequences being compared. Therefore, when searching a whole database the computation time grows linearly with the size of the database.

In this work, we introduce our new and fast alignment method, which is called ABS (Alignment By Scanning), to provide an approximate alignment of two DNA sequences. We compare our method with the first widely used heuristic program for database similarity searching, 'FASTA' [1], and show that our method provides up to 76% (in average) improvement of the alignment score in comparison to the score provided by the FASTA Algorithm. In addition to that, the ABS Algorithm performs the alignment much faster than the FASTA Algorithm as explaind in the next sections. We also compare the alignment score of the ABS and the FASTA Algorithms with the score of Needleman-Wunsch Algorithm to show how the heuristic ABS alignment is close to the optimal Needleman-Wunsch alignment.

To evaluate the alignment method, we compute the score of alignment using a regular gap penalty for different lengths of sequences. The score of the alignment is computed by adding up the score of each pair of letters which may be match-, mismatch-, or gap score.

The rest of the paper is organized as follows. In Section II, we present some of the latest previous work. Section III presents our Alignment By Scanning (ABS) Algorithm and its rules to split and align the sequences. In Section IV, we show the algorithm complexity and the memory requirements of the ABS Algorithm in comparison to the Needleman-Wunsch and the FASTA algorithms. Experimental results are presented in Section V and concluded in Section VI.

II. Related Work

A number of different methods were introduced to align DNA, Protein, or both sequences. The first widely-used efficient global heuristic alignment program was MUMmer [7]. It uses suffix trees to identify maximal unique matches (MUMs) between the two sequences. BLAT program [8] efficiently produces indexing structures that allow one to index whole chromosomes to find seeds, and by its simple change, it allows much higher sensitivity. LAGAN [9] is similar to the BLAT approach which builds alignments in an anchor-based fashion. It allows some flexibility in the dynamic programming around the anchors. [10] and [12] reduce the inexact matching problem to the exact matching problem and implicitly involve two steps: identifying exact matches and building inexact alignments supported by exact matches. To find exact matches [10] used suffix tree but [12] used enhanced suffix array. [11] designed algorithm for direct comparison and proposed an alternate scoring scheme based on fuzzy concept.

III. Alignment By Scanning Algorithm 'ABS'

A. Basic Idea

FASTA Algorithm restricts the shifting of whole sequence to be only in one direction, this will result in less number of matching between both sequences because there is no flexibility in shifting the sequences. Increasing the number of matches may be achieved when part of the sequence (we call it subsequence) is shifted in one direction and other part is shifted in other direction. Therefore, splitting the sequence into many subsequences and align each subsequence independent from the other increases the chance to have more matches. For that, we introduce fast alignment method to provide an approximate alignment of two DNA sequences. We call our algorithm 'ABS' (Alignment By Scanning) because it scans the two sequences simultaneously from the

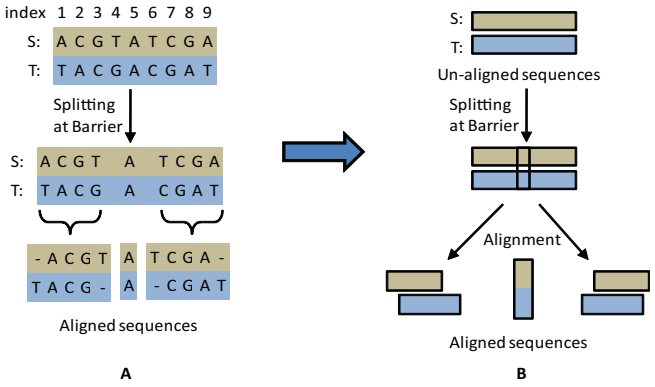

Fig. 1. A: ABS alignment example. B: ABS block diagram. ABS splits the sequences into subsequences at the Barrier. Each subsequence is aligned separately

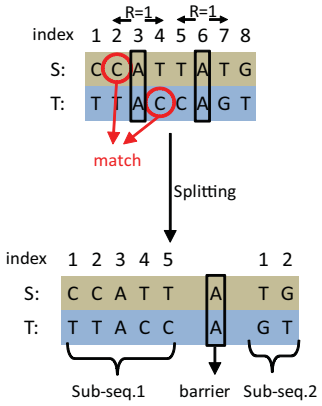

Fig. 2. An example for determining the barrier to split the sequences

beginning till the end to find an approximate alignment and to compute its score.

The ABS starts from the beginning of the sequences (at position 1) and scans the sequences to find a matching between them. The position at which there is a match between S and T creates a split (we call it barrier). This barrier splits the sequence into two subsequences. Each subsequence is aligned independent from the others but in parallel with them. During the scanning, the alignment is done as it is in the FASTA Algorithm by computing the relative positions of each character in the two sequences and finding the highest repeated offset which refers to the amount and direction of shifting.

Fig. 1 shows an example of aligning two sequences using our alignment technique 'ABS'. The ABS scans the two sequences S and T starting from position 1 and continues forward till the end. In the same time, it computes the relative positions and the offsets. At the position 5, it finds that there is a match between S and T with letter A. Therefore, the ABS splits the sequence into two subsequences, aligns the the first subsequence, and continues scanning the the second subsequence. At the end of scanning, both sequences will be aligned and the barrier A will be inserted between them.

B. Rules of Splitting the Sequences

Splitting the sequences into subsequences is based on the way of selecting the positions of the barriers. In the ABS, we do not consider every match in both sequences as a barrier which splits the sequences into subsequences. Instead, we select the barrier such that there is no relation between the letters of the sequence S before the barrier and the letters of the sequence T after the barrier and vice versa in range of 'R' searched letters before and after the barrier. The parameter 'R' is given in the beginning of the alignment and is based on the score of match, mismatch and gap. If this condition is not met, we ignore the barrier and do not split the sequence at that position.

Fig. 2 shows an example how to find the barriers at which the sequences will be split. In this example, we consider that the range of searching before and after the barrier is 1 letter, i.e. R=1. By scanning the sequences, we notice that there are two matches (at positions 3 and 6). For the match at position 3 (letter 'A'), there is relation between S and T around the match 'A' because the letter at position 2 of sequence S and the letter at position 4 of sequence T are the same (which is 'C'). For that, we do not consider the match 'A' at position 3 as barrier. According to this rule, we may consider the second match, which is at position 6, as a barrier because there is no relation between S and T around it in a range of 1 letter. In this case, the sequences S and T will be split into two subsequences at position 6 and each subsequence will be aligned independent from the other.

If the ABS could not detect any match during the scan, it creates a virtual barrier after a certain number of characters, 'L', which is given as input parameter. At this virtual barrier the sequences S and T are split into two subsequences. The condition to select the virtual barrier is the same as the real barrier, i.e., there should be no relation between S and T around the virtual barrier. If this condition is not met after L characters, the next characters are checked until the condition is met. To compute the score of alignment, we use:

$$\begin{aligned} \textit{Alignment Score} \;=\; & (m \times \textit{match\_score}) \\ & + (2 \times s \times \textit{gap\_score}) \\ & + (d \times \textit{mismatch\_score}) \end{aligned} \tag{1}$$

such that, m is the number of matches. s is the number of shifts $\Rightarrow$(2 x s) is the number of gaps. d is the number of mismatches. The scores of the gap, 'gap_score', and the score of the mismatch, 'mismatch_score', have always negative values. If the length 'L' of the subsequence is known, then the number of mismatches 'd' can be computed as:

d = L - m - s.

Finding the highest score of alignment is based on the scores of match, mismatch, and gap. That means, having more matches not always means getting the best alignment score because more gaps and mismatches may be added and these will decrease the score of alignment. For that, the ABS gets the scores of match, mismatch and gaps as input parameters and computes the score of alignment for

Algorithm	Algorithm Complexity	Memory Requirments
Needleman-Wunsch	O(m x n)	(m x n)
FASTA	O(w x (m + n)	(m + n + (w x (m + n)))
ABS	O(min(m,n))	(2 x (m + n) + 1)

Fig. 3. Comparing the complexity and the memory requirements between the Needleman-Wunsch, the FASTA and the ABS

each possible alignment, then it selects the alignment which achieves the best score.

IV. Algorithm Complexity and Memory Requirements

In this section, we compare between the complexity and the memory requirements of the Needleman-Wunsch Algorithm, the FASTA Algorithm, and the 'ABS' Algorithm.

In case of the Needleman-Wunsch Algorithm [6], let *m*, *n* are the lengths of the query sequence 'S' and the database sequence 'T', respectively. As the Needleman-Wunsch Algorithm is based on dynamic programing, then its complexity is $O(m \times n)$.

The memory requirement of the Needleman-Wunsch Algorithm based on the length of the sequence S and T, i.e, m and n. Therefore, its memory requirement is $(m \times n)$.

In case of the FASTA Algorithm [1], let *r* is number of sub-alignments. and let *w* is the width of the restricted area in which the dynamic programing is applied such that w << n. The complexity of the FASTA Algorithm is $O(w \times (m+n))$. The memory requirement of the FASTA Algorithm based on the length of the sequence S and T and the width 'w' of the restricted area in which the dynamic programing is applied. To store the indices of each character of the sequences S and T, we need *m* and *n* memory cells, respectively. To apply the dynamic programing on the restricted area, we need $(w \times (m+n))$ cells. Therefore, the memory requirements of the FASTA Algorithm is $(m + n + (w \times (m + n)))$.

In case of the ABS Algorithm, the algorithms scans the two sequences together and splits them into many sub-sequences during the scan. So, its complexity is $O(min(m, n))$. Therefore, the ABS Algorithm is much faster than the FASTA Algorithm.

To store all possible offsets in the range [-m to n], we need m + n + 1 cells. Therefore, the memory requirement of the ABS Algorithm is $(2 \times (m + n) + 1)$.

Fig. 3 shows summary of comparing the algorithm complexity and the memory requirements between the Needleman-Wunsch, the FASTA and the ABS.

V. Experimental Results

The evaluations are conducted using different lengths sequences of HUMAN division downloaded from the well known DNA sequences of the database "DNA Data Bank of Japan" (ddbj) [13]. Figure 4 shows the accession number of the sequences which are used for our evaluations. The sequences are divided into four different lengths, 50, 100, 200, and 500 nucleotides. The first row of the table presents

Length of 50	Length of 100	Length of 200	Length of 500
GX386459	GX389453	HQ032039	FQ377774
GX713604	GX689428	GX523452	GU758572
GX714613	GX687421	GX862587	GU532615
GX716623	GX687420	GX689434	HM669508
GX390496	GX680445	GX536532	HM668499
GX714618	HI955819	GL688802	GU754539
GX713601	GX672374	GU811161	GU908783
GX715614	GX285798	GX405587	GU511336
GX390494	HQ662280	GX512352	GU517380
GX390497	GX871623	GX650226	GU521405
GX713600	AF501247	GL688805	GU537666

Fig. 4. The accession number of the sequences used for our evaluations. The first row shows the query sequences. the remaining rows show the database sequences

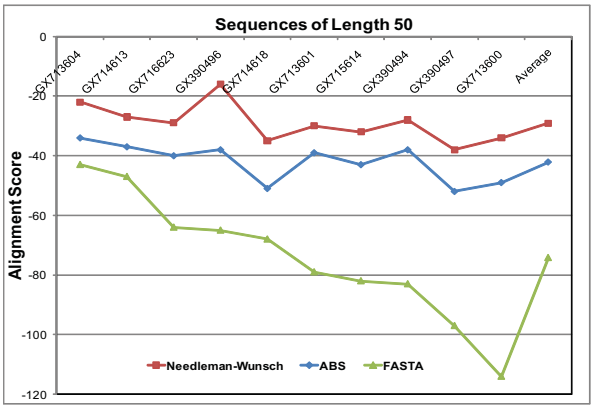

Fig. 5. Results of aligning sequences of length 50 using Needleman-Wunsch, ABS, and FASTA Algorithms

the query sequence for each different length. The remaining rows shows the database sequences.

We compare our method with the first widely used heuristic program for database similarity searching FASTA. To do that, we align the query and the database sequences which are presented in Figure 4 using FASTA35 program [5]. This program is slightly modified to align the sequences globally. The option "-a" of the FASTA35 program is used to to align all of both sequences. The k-tuple parameter of the FASTA35 is selected to be '1' to find the best alignment regardless the time required for alignment.

We also compare our method with the Needleman-Wunsch Algorithm to show how our heuristic ABS alignment is close to the optimal Needleman-Wunsch alignment. The score of match, mismatch or gap are selected to be the same in all three programs (ABS, FASTA35, and Needleman-Wunch).

The experimental results are presented in Figures 5 - 8. Each figure shows the score of alignment for the Needleman-Wunsch, the ABS, and the FASTA Algorithms. The last column in each figure shows the average alignment score for the different programs through all the database sequences.

Figure 5, Figure 6, Figure 7, and Figure 8 show the results for database sequences of length 50, 100, 200 and 500 Nucleotides, respectively.

In all figures, the alignment score of the ABS Algorithm is better than the score of the FASTA Algorithm for all database

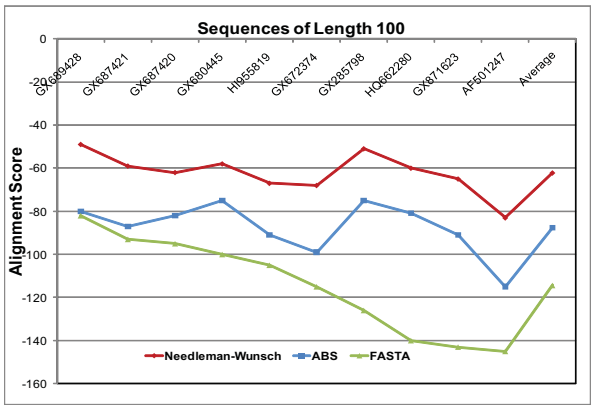

Fig. 6. Results of aligning sequences of length 100 using Needleman-Wunsch, ABS, and FASTA algorithms

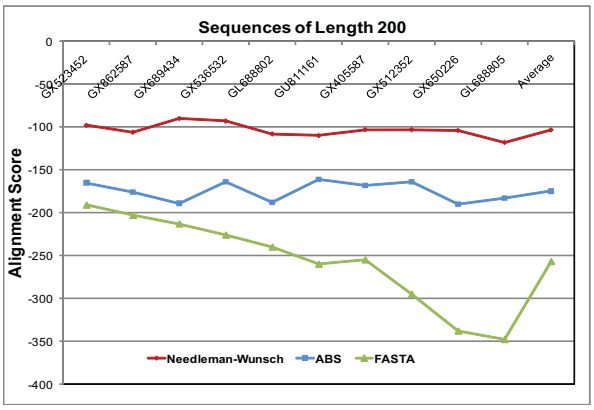

Fig. 7. Results of aligning sequences of length 200 using Needleman-Wunsch, ABS, and FASTA algorithms

sequences. For example, the alignment score of the ABS better than the score of FATSA on average of 76%, 30%, 50%, and 37% for the sequences of length 50, 100, 200, and 500 Nucleotides, respectively.

The figures also show how far is the result of the heuristic algorithms (ABS and FASTA) from the result of the optimal one (Needleman-Wunsch). The score of the ABS is much closer to the score of Needleman-Wunsch in comparison to the score of the FASTA for the same sequence length. For example, in the case of length 50 Nucleotides, the average alignment score of the ABS is 44% far from the optimal score of the Needleman-Wunsch but the FASTA is 154% far from the Needleman-Wunsch. This shows that the alignment score of the ABS is more than 100% closer to the optimal alignment score of Needlman-Wunsch Algorithm in comparison to the FASTA Algorithm.

On the other hand, the processing time of our ABS Algorithm is much faster than the FASTA and the Needleman-Wunsch algorithms (as discussed in Sec. IV). For example, considering m and n are equal to 500, then the complexity of the Needleman-Wunsch will be O(250000). Considering that the width of the restricted area w=10, then the complexity of the FASTA will be O(10000). As both sequences have the same length. then the complexity of the ABS algorithm will be O(500).

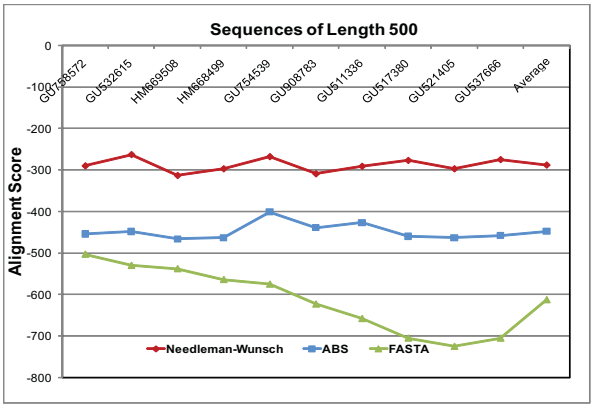

Fig. 8. Results of aligning sequences of length 500 using Needleman-Wunsch, ABS, and FASTA algorithms

VI. Conclusion

We have presented new sequence alignment algorithm to provide approximate alignment in comparable time. Our technique splits the sequences at special positions into many subsequences and then aligns each subsequence independently To measure the efficiency of our algorithm we compared it with the FASTA Algorithm and showed that our algorithm achieves better results which are 3.5x closer to the optimal alignment score when it is compared with the FASTA Algorithm.

References

[1] European Bioinformatics Institute Home Page, FASTA searching program, 2003. http://www.ebi.ac.uk/Tools/sss/fasta/.

[2] T. F. Smith and M. S. Watermann. Identification of common molecular subsequence. Journal of Molecular Biology, 147:196-197, 1981.

[3] Talal Bonny, M. Affan Zidan, and Khaled N. Salama: An Adaptive Hybrid Multiprocessor Technique for Bioinformatics Sequence Alignment. The 5th Cairo International Conference on Biomedical Engineering Conference, (CIBEC10), pp.112-115, December, 2010.

[4] M. Affan. Zidan, Talal Bonny, and Khaled N. Salama: High Performance Technique for Database Applications Using Hybrid GPU/CPU Platform. Proceedings of the great lakes symposium on VLSI (GLSVLSI'2011), pp 85-90. May 2-4, 2011.

[5] http://fasta.bioch.virginia.edu/fasta_www2/.

[6] Needleman, S. and Wunsch, C. A general method applicable to the search for similarities in the amino acid sequence of two sequences. Journal of Molecular Biology, 48(3), 443-453, 1970

[7] Delcher, A. L., S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. Nucl. Acids Res. 27, pp. 23692376, 1999.

[8] Kent, W. J. BLATthe BLAST-like alignment tool. Genome Res. 12 (4), pp. 656664, 2002.

[9] Brudno, M., C. Do, G. Cooper, M. Kim, et al. LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA. Genome Res. 13, pp. 721731, 2003.

[10] Abouelhoda MI, Kurtz S, Ohlebusch E. Replacing suffix trees with enhanced suffix arrays. Journal of Discrete Algorithms, 2: pp. 53-86. 2004

[11] Subhra Sundar Bandyopadhyay, Somnath Paul and Amit Konar, Improved algorithm for DNA sequence alignment and revision of Scoring matrix, Proc. Of ICISIP 2005.

[12] Hoffmann S, Otto C, Kurtz S, et al. Fast mapping of short sequences with mismatches, insertions and deletions using index structures. PLoS Comput Biol 5:e1000502, 2009

[13] http://www.ddbj.nig.ac.jp/