Time and Space Efficient Optimal Pairwise Sequence Alignment using GPU

Ardhendu Sarkar*, Kinjal Ray†, Debaroti Chowdhury‡, Kishan Sahu§, Solanki Kundu¶, Surajeet Ghosh‖
Dept. of Computer Science & Technology, Indian Institute of Engineering Science & Technology, Shibpur, Howrah, India
Email: *ardhendu.rs2018@cs.iiests.ac.in, †kinjalray.ug2017@cs.iiests.ac.in, ‡debarotichowdhury@gmail.com,
§kishansahu6249@gmail.com, ¶solankikundu20.1999@gmail.com, ‖surajeetghosh@ieee.org

***Abstract*—Sequence alignments are currently gaining close attention due to their great impact on the quality aspects of life such as facilitating early disease diagnosis, identifying the characteristics of a newly discovered sequence, etc.. With the rapid growth of genomic data, searching for a sequence homology over huge databases is unable to produce results within a realistic time. Hence, it demands an efficient sequence alignment accelerator to improve the performance of the system. Though some popular acceleration platforms, like supercomputers, Very Large Scale Integration (VLSI) chip, Graphics Processing Unit (GPUs) and Field Programmable Gate Arrays (FPGAs) based devices are available, but they are unable to meet the intended requirement to the current growth of genome database. This paper presents a parallel dynamic approach for global pairwise alignment algorithm using GPU to handle such large database. The approach performs the alignment procedure in CUDA-enabled GPU platform by creating dynamic threads to perform parallel execution. This GPU-based implementation is tested using pseudorandom database and various length of sequences. The execution result shows reasonably better performance in terms of time and space requirement with respect to existing GPU-based optimal pairwise sequence alignment approaches.**

***Index Terms*—Bioinformatics; Pairwise Sequence Alignment; GPU-based Sequence Alignment**

I. Introduction

Sequence Alignment (SA) is a major part of Bioinformatics and Computational Biology (BCB) which is the intersection part of Biology and Computer Science. Computational biology is using and developing a computer algorithm and architecture for examining biological data. Nowadays BCB is one of the fastest growing hybrid fields in the educational field that creates different computational tools, databases, and methods to support biological research. SA is a basic mechanism in molecular biology for scanning sameness between sequences. Presently, the SA are gaining a great attention due to its strong application such as drug engineering, diagnosis of the disease, newly discovered sequence, pharmaceutical development, criminal forensics, constructing evolutionary trees, finding specific sequences, determining the origin of a sequence, finding homology, quantify the phylogenetic distance between two sequences, identify polymorphisms, mutations between sequences and many more.

Sequences can represent as the string of alphabets.

- Nucleic Acid: Both the nucleic acid DNA and RNA are consist of four bases. Adenine (A), Guanine (G), Cytosine (C) and Thymine (T) are the bases of DNA and Adenine (A), Guanine (G), Cytosine (C) and Uracil (U) are the bases of RNA.
- Amino Acid: Proteins developed by following 20 amino acid - A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, and Y.

If the two genomic sequences are similar and they can share a common ancestral sequence, they are homologous. The dissimilarity of sequences is the result for the evolution of the mutation of nucleotide and amino acid sequences. These mutations can be two types: (i) As a replacement for a symbol by another, and (ii) Insertion and deletion of one or more residues. By which comparison of sequences become more complex because only by positioning we can not compare the sequences. Therefore needed a computational procedure to resort for sequencing alignment, keeping the same order. At the time of processing, sequence alignment will consist of introducing gaps to get a maximum identical character (optimal alignment) between the two sequences. Firstly, local and global sequence alignment are presented in [1] and [2] respectively. Basically, global and local sequence alignment is based on similarity portion of sequence length. Pairwise and Multiple sequence alignment are the two type of sequence alignment based on the number of sequences. Fig. 1 gives an idea of the different types of sequence alignment. This is the general idea of sequence alignment and similarity and for which various algorithm is proposed earlier.

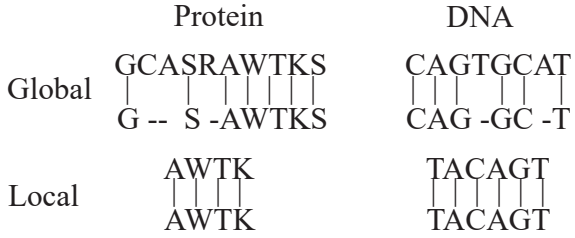

Fig. 1. Global and Local Sequence Alignment of Protein and DNA Alignment

Efficient SA is a significant and demanding task in bioinformatics. It is identical to string matching and used to check the genetic evolutionary correlation over a set of DNA sequences or protein (molecular sequences). Correct alignment can deliver valuable facts for experimentation on the newly query sequences. Till now, many popular SA algorithms and methods have been suggested to conduct sequence alignment

978-1-7281-1895-6/19/$31.00 ©2019 IEEE

tasks, like dot plot [3], Needleman-Wunsch (N-W) using GPU [4], Smith-Waterman (S-W) [1], FASTA [5], BLAST [6], HMMER [7] and ClustalW [8].

Some algorithms are highly optimal after sequence alignment but take maximum computational time. Its computational charge makes it unsuitable for actual purposes. Gradually growth of genomic data, in search of homology sequences over huge genetic databases is unable to produce results in time, hence need for acceleration. To get efficient and optimal sequence alignment results, the SA method has been performed on supercomputers, other parallel architectures and various emerging accelerator platforms such as the very large scale integration chip for special purposes (VLSI), Field Programmable Gate Arrays (FPGAs), Network-On-Chip (NOC), CAM, Cell Broadband Engine, Xeon-Phi and Graphics Processing Units (GPUs) etc.. However, there is always a compromise between area, speed, power, cost and development time and reuse when selecting the algorithm for an acceleration platform. GPUs generally offer more parallelization and higher performance. Additionally, GPUs can also accelerate the sequence alignment procedure. This paper aims to present a GPU-based sequence alignment method over dynamic programming and comparison over different dynamic programming methods.

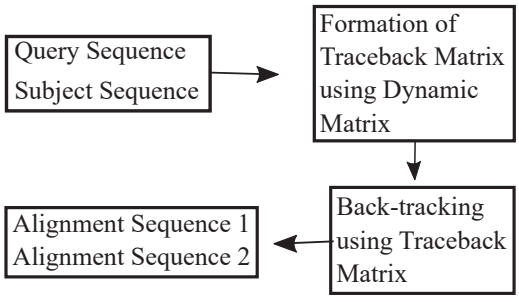

Fig. 2. Block diagram of the proposed technique

The main purpose of this article is to contribute a sequencing procedure used for aligning sequences of the biological molecules. The objective of this article is to present details about the proposed algorithm tool and to provide a comparison with different techniques. In the fast-growing world of BCB, new tools used for sequence alignment is substituting the older tools. Hope, this algorithm article help the researchers of the field of BCB.

In dynamic programming approach time complexity, space complexity respectively O(nm), O(nm), respectively, where n and m are the length of two sequences. With our proposed method, both the complexities can be minimized to O(n). Time can be reduced with the anti-diagonal parallel approach depends on the size of the matrix. The space can also be reduced by not keeping the whole record of the matrix which is no longer required for final sequence alignment but keeping only current and the previous two anti-diagonal row of the matrix for cell value calculation. This article describes parallelization for fill up of the dynamic matrix in an anti-diagonal approach. This algorithm focuses on parallelization to reduce time and space using CUDA-enable GPU. Firstly, in Section II explains parallelization approach used in an anti-diagonal way. In Section III a dynamic alignment algorithm is developed using CUDA C with the help of GPU hardware. In Section, IV describes the performance evolution of the proposed method. Section V summarizes the paper by a brief conclusion.

II. Parallel Approach in GPU

In the last few years, the use of Graphics Processing Units (GPUs) have enlarged to enhance the performance of computational units. Recent GPUs are capable of delivering a very complex assignment with massive performance enhancement. Day by day, GPU is emerging as powerful hardware for computing capability over general purpose CPUs (Central Processing Units). GPU has high performance as it executes tasks parallelly. Utilizing GPUs capability for general purpose programming is expanding with time after new programming interface Compute Unified Device Architecture (CUDA) is developed by Nvidia for Nvidia GPUs. Mainly GPUs was developed for high graphics purpose, but now it is used for the non-graphics applications for its immense computing capability. In the dynamic programming method for sequence alignment, the alignment operation is performed using 2-dimensional matrix where each cell value is based on the three values: left (L), up (U) and diagonal (D) value.

The matrix values could be generated by either row-wise, or column-wise or by anti-diagonal-wise. For parallel generation of cell values, anti-diagonal method would be most favourable approach. Fig. 3 depicts how matrix cell values are generated in parallel manner using anti-diagonal method. It is very much clear from the figure that, one iteration can produce a number of cell values concurrently. The row-major order of computation in each iteration is shown in Fig. 3 for a (8×8) matrix. In first, second, third iteration position {(1, 1)}, {(1, 2) and (2, 1)}, {(1, 3), (2, 2) and (3, 1)} will be evaluated respectively. The cell values for every iteration will be evaluated in equidistant. This process to be continued up to the right lower position. It is obvious from Fig. 3 that in the eighth iteration supreme parallelism can be achieved. Every value of a row depends on the previous two rows of Fig. 3 from the third row and all the value can be achieved in a parallel manner. Therefore, the number of matrix rows is reduced to 3 which is irrespective to the size of sequences. In the proposed algorithm, the following behaviour of various phases have been observed.

- **Initialization:** This phase deals with first and second row and are always initialized. All the other intermediate cell value is computed from two previous rows.
- **Growing Phase:** Third to eight rows are involve in this phase. Leftmost and rightmost cell values of every row are calculated from the leftmost and right most cell values. An intermediate cell position (i, j) is computed considering from the cell positions $((i-2),(j-1))$ as diagonal, $((i-1),(j-1))$ as left, $((i-2), j)$ as up.

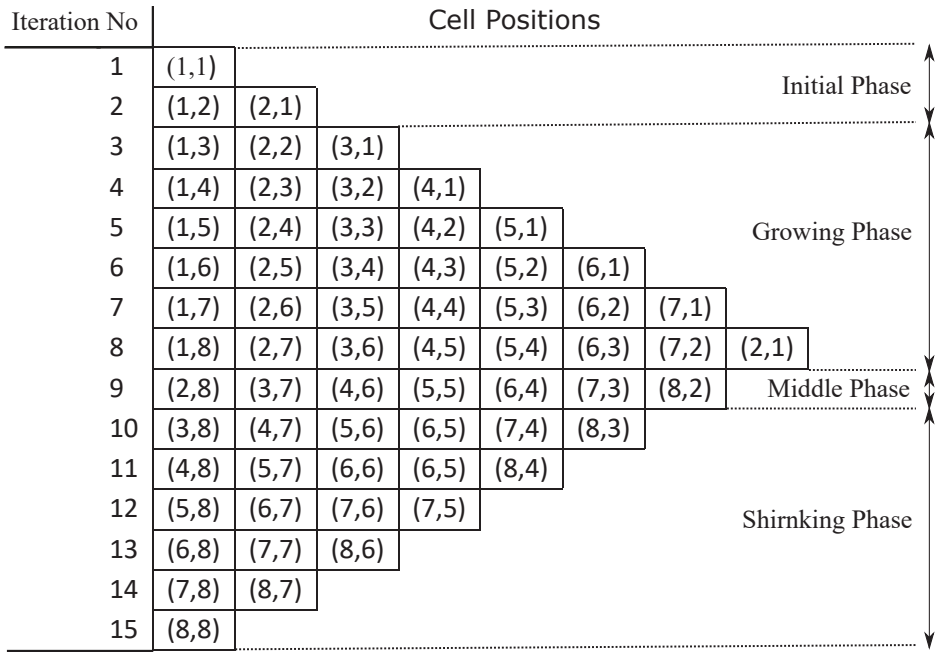

Fig. 3. Cell value generation in different iteration process

- **Middle Phase:** This phase includes the ninth row. An intermediate cell position (i, j) is computed considering from the cell positions $((i-2), j)$ as diagonal, $((i-1), j)$ as left, $((i-1), (j+2))$ as up.
- **Shrinking Phase:** From tenth to fifteen rows are participated in this phase. An intermediate cell position (i, j) is computed considering from the cell positions $((i-2), (j+1))$ as diagonal, $((i-1), j)$ as left, $((i-1), (j+1))$ as up.

In the next section, the proposed algorithm is illustrated the parallel approach for computing the sequence alignment. Here cell value elements of different rows are computed concurrently.

III. Proposed Alignment Algorithm

In this technique, the optimal pairwise global alignment between two DNA sequences is proposed. To perform the alignment of sequences, a foundation matrix (FM) of size $3\times((n$ or $m)$+1) and a traceback matrix (TM) of size $(n+1)\times(m+1)$ are used, where n, m are the length of sequences. The lower value of m and n is the column size of FM. The Fig. 2 reveals the procedure of the proposed alignment technique. The procedure of pair-wise sequence alignment requires three remarkable functions.

- Initialization of the TM and FM.
- Generation of Traceback Matrix using Foundation Matrix.
- Aligned sequence formation using Traceback Matrix.

Alignment procedure from Traceback matrix is the same as the back-tracking method of [9] paper. Generally, the last procedure applied for all the processes used for dynamic programming.

A. Initialization of FM and TM

The foundation matrix and the traceback matrix are used to perform the sequence alignment procedure. Each cell value in the FM is calculated by considering the maximum cell value surrounded by three adjacent cells. TM records a certain predecessor among the three adjacent cells from where the maximum score value (same as Algorithm 2) is produced. This accounting is accomplished by considering three movement direction values namely, 1: up (U), 2: diagonal (D) and 3: left (L) movement, to particularly decide the specific forerunner in the course of the endmost alignment. Here, the method has initialized the gap value (GV) and mismatch value as -1, and match value (MV) as 1. The first and foremost operation for alignment mechanism is to initialize the arrays. TM is formed by considering a $(n+1)$ by $(m+1)$ array and n nucleotides sequences, were filling up of every array cell necessities three adjacent array cells, i.e., up, diagonal and left. The formation of the TM in terms of direct pointers is entirely dependent on the cell contents of FM. Starting position of TM $(0, 0)$ is initialized, i.e.; $TM_{0,0}$ = 'E' as shown in Fig. 4. Except starting position, the cell values of the first row are filled by 'L' and the cell values of the first column are crowded by 'U'. The value of the first row's (FP) first element in the FM is initialized to '0'. First two elements of the middle array (MP) value is initialized by 'GV' value. The operation starts from the upper left-hand corner, i.e. cell (0, 0), in TM and progresses until it reaches cell lower right-hand corner $(n+1, m+1)$. The cell in the FM is filled up by considering the maximum array cell value surrounded by three adjacent cells by using three rows, namely, first position (FP), middle position (MP), last position (LP) of the array. In order to

Algorithm 1 : **Proposed GPU-Based Method**

Input: Two Sequences $S_1 = a_1, a_2, a_3...a_n$ and $S_2 = b_1, b_2, b_3...b_n$
Output: Optimal alignment and score SC

FM: Foundation Matrix of size $(3, (n+2))$
where FP: first pointer, MP: middle pointer and LP: last pointer
TM: Traceback Matrix $((n+1), (m+1))$
GV: Gap Value
MV: Match value

1) $FM_{0,0} = 0$; $FM_{1,0} = 0$; $FM_{0,0} = 0$
2) $TM_{0,0} =$ 'E'; $TM_{k,0} =$ 'U' for $k = 1$ to n; $TM_{0,k} =$ 'L' for $k = 1$ to n
3) $FP = 0$ $MP = 1$ $LP = 2$
4) $j = thread\_id + 1$
5) For $i = 1$ to n
   IF $i < n$
   $FM_{LP,0} = FM_{LP,i+1} = (i+1) \times GAP$
   $ii = i - j$, $jj = j - 1$ And $fmc = j$
   IF($S1_{ii} == S2_{jj}$) $SC = MV$
   Else $SC = GV$
   Else if $i = n$
   $ii = i - j$, $jj = j - 1$ And $fmc = j - 1$
   IF($S1_{ii} == S2_{jj}$) $SC = MV$
   ELSE $SC = GV$
   Else
   $ii = n - j$, $jj = i + j - n - 1$ And $fmc = j - 1$
   IF($S1_{ii}$) $== S2_{jj}$) $SC = MV$
   ELSE $SC = GV$
   $FM_{LP,fmc}$=MAX($FM_{MP,fmc}$+GV,$FM_{MP,fmc-1}$ +GV, $FM_{FP,fmc-1}$+SC, $\&PTR$)
   'D'/'U'/'L' set in PTR
   $TM_{ii+1,jj+1}$=PTR
   FP=(FP+1)%3 MP=(MP+1)%3 LP=(LP+1)%3
6) $Traceback\_Method(TM)$

Algorithm 2 : **MAX (D1, D2, D3, *PTR)**

Input: Calculative value of UP (D1), LEFT (D2), DIAGONAL (D3)
Output: $MAX of D1, D2, D3$ And PTR

MAX: Maximum value

1) $(D1 >= (D2\ \&\ D3)$
   $MAX = D1$; $PTR =$ 'U'
2) $(D2 >= (D1\ \&\ D3)$
   $MAX = D2$; $PTR =$ 'L'
3) $(D3 >= (D1\ \&\ D2)$
   $MAX = D3$; $PTR =$ 'D'

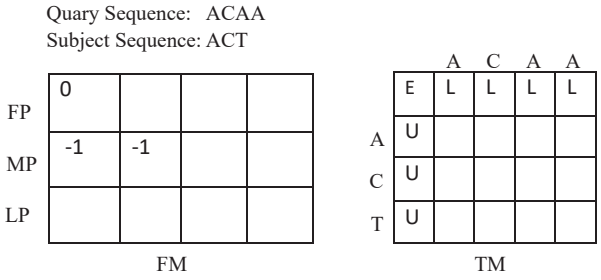

Fig. 4. Initialization of Foundation Matrix and Traceback Matrix

compute the score value of LP array in FM, the score values of its three adjacent cells are compared, viz., the left, the above and the diagonal, which are actually present in FM array. The score of an array cell is calculated with the help of its top and left neighbours and depending on mismatch and match of nucleotides of sequences. Finally, the respective cell in TM is filled with the correct movement value, i.e., with 'U', or 'D' or 'L', from where the maximum cell value was obtained in FM. Formation of the TM cell value is fill up concurrently with the MAX (Algorithm 2) value generation of FM for every iteration process.

B. Formation of TM

This approach uses a $3 \times (m+1)$ foundation matrix instead of $(n+1) \times (m+1)$ matrix as in [9], [10] for better space efficiency. This is an important minimization during the filling up operation of last array position of FM. Filling up of each cell of LP position requires only its two previously filled neighbouring row cell information. Hence, i^{th} row requires information of only its two previous rows, i.e., (i-1) and (i-2). Keeping the MP, LP rows unchanged and over-writing next content to its FP row (see Algorithm 1) made this optimization possible. Let us take an example of 2 DNA sequences (as shown in Fig. 4) to explain the filling up procedure in FM (see Fig. 5(A)). At first, the first row is filled up by value 0 and the first two columns of the second row are filled up based on the initial gap penalties. Now the third-row values can be calculated from the previous rows. Afterwards, the first row (FP) values are no more needed, so it is overwritten by the calculating scores and this row working as an LP, where the second row (MP) and the third row (LP) is working as FP, MP respectively.

A similar process will happen up to $(n + m - 1)$ times where rows can organize in the circular queue from to store the element. As there is no loss of information because at the time of each iteration formation first array is always overwritten by the parallel computed value for an iteration. For the TM matrix, every cell value fills up in parallel manner for each iteration. The TM is filled up in anti-diagonal row (i.e., from upper-left corner to lower-right corner) manner. At the time of MAX value generation (see Algorithm 2), cell value of TM will be calculated concurrently. Although older cell values of FM are overwriting as the procedure moves further, there will be no effect on the algorithm as the sequence alignment is done only on TM. For alignment purpose, it uses only the direction values and not the cell score values. FM matrix will always contain only three rows for any size (depending on the number of nucleotides) of the DNA (RNA or protein) sequences. After

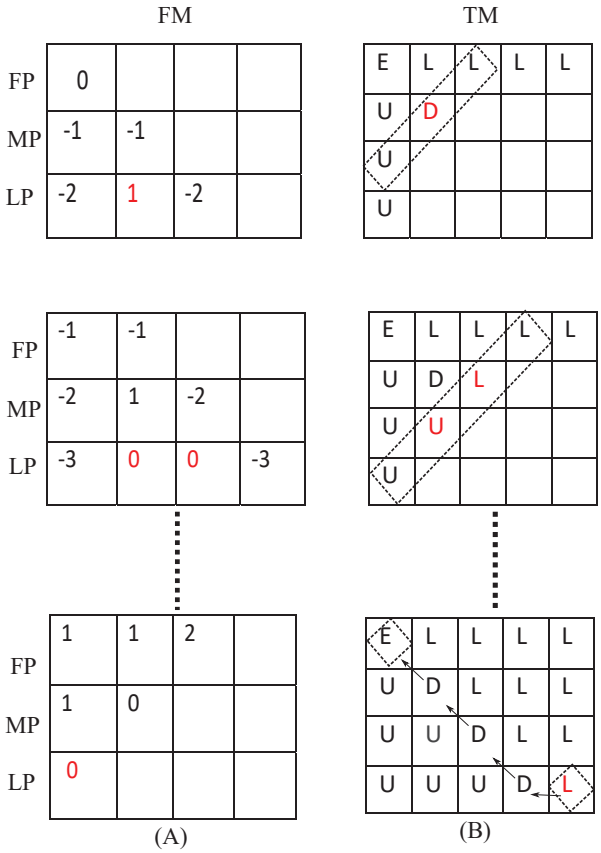

Fig. 5. Foundation of Traceback Matrix

the ending of the iteration, traceback process is started to get the aligned sequences using TM.

C. Alignment using Traceback Matrix

The last stage of alignment procedures begins just after the ending of the iteration process. No data will be inserted into the foundation matrix. Traceback matrix $(n+1) \times (m+1)$ is formed by putting the direction value to the cell. After the sixth operation, the traceback procedure begins for the given sequences (see Fig. 4). For the particular sequence, direction on the trackback procedure is shown in Fig. 5(B). The alignment of the sequences is performed from the last cell of TM. In the previous section we discussed that the movement on TM is depended on the cell value of the TM. That direction cell value has three directions (viz., left, up and diagonal). One of the neighbouring cells is selected out of the three direction cell. Vertical movement, diagonal movement, horizontal movement happens if the cell value is found to be 'U', 'D' and 'L' respectively. Alignment of the sequences starting from the end nucleotide of both sequences. Gap '–' is placed in the sequences for the different direction cell value. For every movement, a gap value or a nucleotide is placed to the aligned sequence.

- Vertical Movement: Gap '–' is initialized in subject sequence and corresponding query sequence nucleotide is placed.
- Diagonal Movement: The corresponding nucleotide associated with sequences, i.e., (query and subject sequence) are placed to the aligned sequences.
- Horizontal Movement: Initialized a gap in the alignment of the first DNA strand in query sequence.

The similar procedure is followed until it reaches the position (0, 0) cell of the traceback matrix. For the different movement, different score is added to the score (SC) value to get the probability of sameness between the sequences at the time of alignment process and finally reverse the aligned strings. Here aligned query and subject sequences are respectively 'ACAA' and 'ACT–' (see Fig. 4) for the ACAA (query) and ACT (subject) sequences (see Fig. 5).

IV. Perfomance analysis

The working principle of the proposed parallel dynamic programming based sequence alignment algorithm method is simulated in Intel-based CPU CORE i7 3.40GHZ machine. To improve the performance of the proposed process, the computational time has been observed in the NVIDIA GeForce GT 710 GPU platform in CORE i5 3.20GHZ processor. The proposed approach is implemented in different data sets of various lengths of sequences varying from 16, 32, 64, 128 and 256. Similar length is used for the query and targeted sequences in experimented time. Each data set is a collection of dissimilar pseudorandomly generated sequence and this set of dissimilar input sequences were examined against various scheme such as [2], [11], [9], [12] with proposed method in Table I. Time and Space complexity for the previous dynamic programming approach are O(nm) and O(mn) respectively, where m and n are the lengths of the two sequences. O($n+m$) is the time complexity of traceback matrix formation for two DNA sequences of size n and m nucleotides. The proposed sequence alignment technique requires O($n+m$) time complexity to align two n and m nucleotides DNA sequences because it depends on number of anti-diagonal row of $(n \times m)$ matrix.

In the proposed scheme, though two matrices are used, viz., foundation matrix of $3 \times ((m \text{ or } n)+1)$ and traceback matrix of $(n+1) \times (m+1)$, yet the required memory space is less than previous approaches as the foundation matrix uses 3 rows and traceback matrix cell uses 2-bit value for backtracking. Hence, the required space for the foundation matrix is of O($3(n$ or $m)$) = O(n or m) and traceback matrix is of O(nm) with 2-bit direction value for each cell.

A comparative analysis of alignment time taken by the GPU over different alignment algorithms is shown in Table I. It is evident from the table that the presented GPU-based scheme consumes reasonably less amount of time for processing. A throughput study has been presented by graphical representation in Fig. 6. Figure depicts the throughput of various alignment approaches in terms of Kilo Sequences Per Second

(KSPS). It has been observed that the throughput has been increased in the presented approach.

In GPU-based computation analysis, data transfer time from host to the device and a device to host are important parameters. In both the cases, the proposed method consumes reasonably less amount of time than [12] method. A comparison of data transfer time has been shown in Table II. The proposed method took less time for data transfer between the host and device, because the method uses foundation matrix $3\times((m \text{ or } n)+1)$ instead of dynamic matrix $(n+1) \times (m+1)$. While increasing the sequence length, the difference of data transfer time between the proposed and [12] method increases and the alignment time is also enhanced.

TABLE I
TIME COMPARISON (IN μS) OVER DIFFERENT ALIGNMENT ALGORITHMS

Sequence Length	GPU activities time (in μs)				
	[2]	[11]	[9]	[12]	Proposed
16	190.63	274.05	190.21	50.56	48.929
32	330.88	690.41	363.62	100.64	97.28
64	914.38	2375.9	1075.5	192.13	183.97
128	2753	8694.2	3434.8	375.14	356.48
256	9990	33541	12675	791.88	710.73

TABLE II
DATA TRANSFER TIME (IN μS) BETWEEN DEVICE AND HOST

Sequence Length	Host to Device		Device to Host	
	[12]	Proposed	[12]	Proposed
16	5.696	4.288	3.392	1.632
32	7.712	6.208	3.776	1.856
64	13.024	8.064	12.736	6.208
128	84.545	43.585	70.145	34.344
256	335.52	158.21	297.25	149.28

V. CONCLUSION

An effective GPU-based sequence alignment scheme has been presented in this paper which can align sequences of variable length of inputs. This proposed approach is memory and time efficient compared to the well known different approaches. This approach also preserves the accuracy of alignments. The experimental result proves that the proposed scheme runs significantly faster and produces remarkable performance than the other traditional existing schemes. The GPU implementation of the proposed scheme increases speed up factor of the processing time to a further extent and significant results have been noticed. Due to resource constraints of GPU platform, we couldn't test beyond 256 nucleotide sequences. Future work includes the adaption of spatial parallelism in the proposed method to further improve throughput for large size sequences.

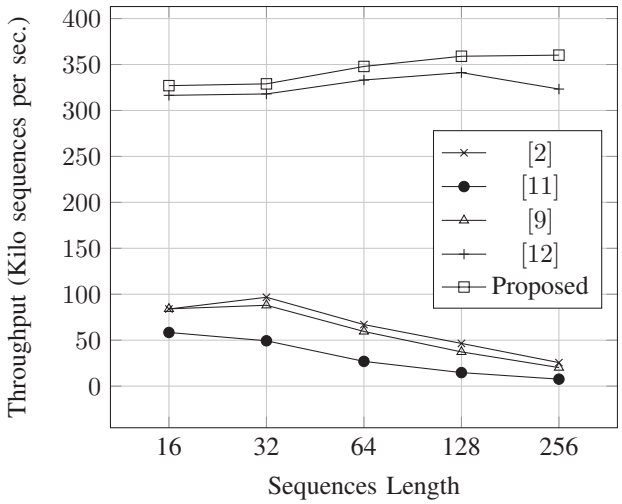

Fig. 6. Throughput analysis of the proposed scheme with respect to existing alignment procedures

ACKNOWLEDGEMENTS

This work is supported under grant no. BT/PR16378/ BID/7/600/2016 by the Department of Biotechnology, Ministry of Science and Technology, Government of India.

REFERENCES

[1] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences", *Journal of molecular biology*, Vol. 147, No. 1, pp. 195–197, 1981.

[2] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins", *Journal of molecular biology*, Vol. 48, No. 3, pp. 443–453, 1970.

[3] A. J. Gibbs and G. A. McIntyre, "The diagram, a method for comparing sequences: Its use with amino acid and nucleotide sequences", *European journal of biochemistry*, Vol. 16, No. 1, pp. 1–11, 1970.

[4] C. S. Khaladkar, "An efficient implementation of needleman wunsch algorithm on graphical processing units", *Honours Programme of the School of Computer Science and Software Engineering, The University of Western Australia*, 2009.

[5] D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches", *Science*, Vol. 227, No. 4693, pp. 1435–1441, 1985.

[6] S. F. 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.

[7] S. R. Eddy, "Profile hidden markov models", *Bioinformatics (Oxford, England)*, Vol. 14, No. 9, pp. 755–763, 1998.

[8] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice", *Nucleic acids research*, Vol. 22, No. 22, pp. 4673–4680, 1994.

[9] S. S. Ray, A. Banerjee, A. Datta, and S. Ghosh, "A memory efficient dna sequence alignment technique using pointing matrix", In *2016 IEEE Region 10 Conference (TENCON)*, pp. 3559–3562, 2016.

[10] S. A. Manavski and G. Valle, "Cuda compatible gpu cards as efficient hardware accelerators for smith-waterman sequence alignment", *BMC bioinformatics*, Vol. 9, No. 2, pp. S10, 2008.

[11] S. S. Ray, N. Srivastava, and S. Ghosh, "A hardware-based high-throughput dna sequence alignment scheme", In *2016 IEEE Annual India Conference (INDICON)*, pp. 1–6, 2016.

[12] A. Chaudhary, D. Kagathara, and V. Patel, "A gpu based implementation of needleman-wunsch algorithm using skewing transformation", In *2015 IEEE Eighth International Conference on Contemporary Computing (IC3)*, pp. 498–502, 2015.