Sādhanā (2019) 44:171
https://doi.org/10.1007/s12046-019-1141-x

MPSAGA: a matrix-based pair-wise sequence alignment algorithm for global alignment with position based sequence representation

JYOTI LAKHANI[1,2], AJAY KHUNTETA[2], ANUPAMA CHOUDHARY[3] and DHARMESH HARWANI[4,*]

[1]Department of Computer Science, Maharaja Ganga Singh University, Bikaner 334 004, India
[2]Department of Computer Engineering, Poornima University, Jaipur, India
[3]Department of Computer Science, Keen College, Bikaner, India
[4]Department of Microbiology, Maharaja Ganga Singh University, Bikaner 334 004, India
e-mail: jyotilakhanimgsu@gmail.com; khutetaajay@poornima.org; chowdharyanupama@gmail.com; dharmesh@mgsubikaner.ac.in

MS received 9 May 2018; revised 7 April 2019; accepted 16 April 2019; published online 29 June 2019

Abstract. The proposed algorithm is a novel matrix-based global pair-wise sequence alignment with a de novo sequence representation. Needleman–Wunsch, noblest, Emboss-Needle, ALIGN, LALIGN, FOGSAA, DIALIGN, ACANA, MUMmer, etc. are few other algorithms that are most commonly used for global pair-wise sequence alignment. Needleman–Wunsch algorithm is one of the most popular algorithms that provides the best possible pair-wise sequence alignment, but the algorithm output is associated with high time and space complexities. To resolve these complex issues, researchers have proposed several algorithms to reduce time and space complexities in the pair-wise sequence alignment. Most of these algorithms provide solutions, but compromise the optimal result in favor of plummeting time and space complexities. An attempt has been made in the present research to develop MPSAGA and a completely unique positional matrix (PM) based sequence representation to deal with the time and space complexities without compromising sequence alignment results (MPSAGA is in public domain available at https://github.com/JyotiLakhani1/MPSAGA). A benchmarking of the proposed algorithm has also been performed with other popular pair-wise sequence alignment algorithms with and without positional matrix-based sequence representation. The use of an integer instead of string data type and exclusive clustering method in MPSAGA with positional matrix based sequence representation resulted in a noteworthy reduction in the memory usage (space) and execution time in the pair-wise alignment of biological sequences.

Keywords. Clustering algorithms; algorithm design and analysis; data mining; dynamic programming.

1. Background

BIOLOGICAL sequences are too large to be stored in the familiar data structures. A biological sequence is a single, continuous molecule of nucleic acid or protein. It can be thought of as a multiple inheritance class hierarchy of the underlying molecule type: DNA, RNA or protein. The biological sequences are gigantic enough to be considered as a big data and there are many biological databases to store such type of data. Biological databases are libraries of life science information, collected from scientific experiments, published literature, high-throughput experiment technology, and computational analysis. Some examples of biological databases are EMBL [1, 2], GenBank [3, 4], DDBJ [5, 6], Entrez [7], etc. Some important biological sequence analysis is pair-wise sequence alignment, multiple sequence alignment, gene search, gene prediction, gene expression analysis, phylogenetic analysis, etc. There are some dedicated computational algorithms for biological sequence analysis. To execute these algorithms one needs to input the sequence/s in question or sequence accession number to the algorithm. These sequences are required to be held in a variable. The string is the most common representation of a sequence of gene, gene name, DNA, RNA or primary structure of a protein sequence. These strings are stored in a flat file format in the system. FASTA [8] format is an example of a flat file. There are two measures to find out the effectiveness of an algorithm. These two measures are the space complexity and the time complexity of the

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12046-019-1141-x) contains supplementary material, which is available to authorized users.

*For correspondence

aforesaid algorithm. The units of space required to hold data in an algorithm is known as its space complexity. The space complexity plays the most prominent role in solving complex alignments. The space complexity in the analysis of biological sequence will increase n-fold with the increase in the size of biological sequences or sometimes when long and complex algorithms are used. The time complexity of an algorithm is the measure of the time required to execute an algorithm. The space and time complexities of biological sequence analysis are usually very high due to the complex nature of analysis and in most of the cases, it is difficult to execute these algorithms on a computer system having only basic configuration. This is the reason why most of the biological databases are available on the servers having high storage and advanced processors. Researchers can apply filters to get the desired data on their computer systems. The same is true for the biological sequence analysis algorithms such as BLAST [9], MEGA [10, 11], FASTA [8], etc. Many of these are available on high capacity servers and researchers can run their queries online on the server itself and subsequently results can be saved.

The present communication discusses the role of a newly developed algorithm (MPSAGA: a matrix-based pair-wise sequence alignment algorithm for global alignment) and a completely unique positional matrix (PM) based sequence representation in reducing the time and space complexities to provide the best possible global pair-wise sequence alignment. Pair-wise sequence alignment is the process to arrange two sequences next to each other in such a way that their similar elements (nucleotides) are juxtaposed. There can be several possibilities to align two sequences. In a sequence alignment algorithm, the match score is calculated for the best alignment. The attempts have been made in the present communication to address the following two main objectives. First is to find the quality of the alignment. It is to mention here that there can be $(2n)!/(n!)^2$ possible alignments of the two sequences of length n. Second is to find the best possible alignment. The proposed MPSAGA provide an un-gapped pair-wise sequence alignment as an output. The global un-gapped pair-wise sequence alignment is required in situations where the exact match between two sequences is required. Gaps are acceptable in the sequence alignment, if exact matches are not required. Un-gapped alignment aligner, search for the exact match in the pair of sequence. For example, un-gapped alignment will be highly useful, if one is searching for a gene needs to be utilized for gene therapy. In addition, un-gapped alignment is also important in searching of a malicious injected code from a software code in the cyber attack. It will be of much use when a data packet enquires about the network address/port address to be delivered. It is also useful for pattern recognition problems where searching pattern is a password, a bank account number, social security number, mobile number, etc. The proposed algorithm is based on a dynamic programming approach [12]. Dynamic programming is a special algorithmic approach that divides a problem into sub-problems and by solving each sub-problem, best alignment results are produced. In the proposed algorithm, a pair of sequence is aligned by calculating Hamming Distance. The algorithm also uses a novel position based matrix representation of nucleotide sequences. Using MPSAGA along with position matrix representation of sequences, for pair-wise sequence alignment, a remarkable decrease in the space and time complexity issues has been observed. One of the reasons for these observations is that the biological sequences in computer programming are considered as strings and all algorithms represent these sequences as a string data type. In MPSAGA, we have used positional matrix representation of biological sequences which converts these sequence strings to integer matrix. Subsequently, working with integers is rather easy because integer is a primitive data type and comparison of two integers can simply be performed using a single computer instruction. On the other side, comparison of two strings is relatively complex because these are compared using the methods in Java software. Hence, it is time consuming and an overhead to CPU. The other reason can be attributed to the use of an evolutionary clustering algorithm [12]. Once all the possible alignments are detected by MPSAGA, these are passed on to the evolutionary algorithm which assess the fitness of these possible alignments and select the good ones in a separate cluster (C1). The match score has been used in the present communication as the fitness criteria. Then the only task remaining is to sort and find the best possible alignment with the highest match score from cluster (C1) by ignoring all the other remaining pair-wise sequence alignments with low match scores. This shortens the burden on CPU for searching all the possible alignments and therefore results in the reduction of the time complexity in pair-wise sequence alignment. The present paper is divided into seven sections. The proposed algorithm and a short review on the research work already done by other authors have been presented in the section 2. In section 3, a detailed discussion on the novel positional matrix sequence representation has been presented. The procedure to use this representation of pair-wise sequence alignment is also discussed in this section. Empirical studies are presented in section 4. The complexity analyses, results and conclusions are provided in sections 5, 6 and 7, respectively.

2. An overview

Smith and Waterman have proposed a classical dynamic programming based local pair-wise sequence alignment algorithm. It was one of the first applications of dynamic programming to compare biological sequences [13]. A dynamic programming algorithm with quadratic running time for the same problem (no gap penalty) for proteins was firstly introduced by the Needleman–Wunsch [14]. It is still

widely used for optimal global alignment, particularly when the quality of the global alignment is of utmost importantance. Both of these algorithms represent sequences as a string and then a similarity matrix is used to find out the best possible sequence alignment. One sequence string represents rows of the matrix and the other sequence string represents columns of the similarity matrix. The algorithms scan each and every base pair and find the best possible alignment of the two sequences. However, the algorithms are expensive and complex with respect to the time and space that are directly proportional to the product of the length of two sequences under investigation. This is the reason that why both are not suitable for the analyses of long biological sequences. Recent developments in this field have focused on improvement of the time outlay of the tested algorithm. For example, a Fast Optimal Global Sequence Alignment Algorithm (FOGSAA) was suggested [15]. In this algorithm, the alignment of nucleotide or protein sequences was faster than the other optimal global alignment algorithms, including Needleman–Wunsch algorithm. The paper claims that when compared to the Needleman–Wunsch algorithm, FOGSAA achieved a time gain of 70–90% for highly similar nucleotide sequences (with $>$ 80% similarity), and 54–70% for sequences having 30–80% similarity. While most of the standard alignment methods rely on comparing single residue and impose gap penalties, DIALIGN [16] executes, pair-wise and multiple sequence alignments by comparing entire segment of biological sequences with no gap penalties. MUMmer [17–19] was developed for aligning entire genome sequences rapidly. As an accurate alignment tool, ACANA [20] is also useful in comparative sequence analysis to identify conserved functional regulatory elements. The present details also include BLAST and its variations. The BLAST algorithm converts a nucleotide sequence in words, aligns it with other sequences, calculates match scores, and if the match score of the sequence is less than the threshold value, it discards the sequence. It is important to note that the blast algorithm does not analyze the best possible sequence alignment, but rather gives an optimal sequence alignment with the highest score. Other than the algorithms discussed above, some visualization techniques have also been developed that compares two sequences in their pair-wise alignment by manual methods. DotPlot [21] matrix is such kind of visualization method. Dotter (http://sonnhammer.sbc.su.se/Dotter.html) or Dotlet (http://www.isrec.isb-sib.ch/java/dotlet) can also be used to create dot plots to compare alignment of two sequences. Some other pair-wise sequence analysis tools are AlignMe [22–24], Bioconductor [25–27] and DpAlign (http://search.cpan.org/dist/BioPerl/Bio/ Tools/dpAlign.pm). EMBOSS Needle (https://www.ebi.ac.uk/Tools/psa/emboss_ needle/nucleotide.html) and EMBOSS Stretcher (https://www.ebi.ac.uk/Tools/psa/emboss_stretcher/nucleotide.html) are other online tools that allow analyses of larger sequences in global pair-wise alignment.

2.1 *The algorithm*

An attempt has been made in the present work to perform the global un-gapped sequence alignment of two nucleotide sequences by using the proposed algorithm (MPSAGA) (figure 1). The algorithm is a combination of naive and dynamic programming methods of pair-wise sequence alignment. The proposed algorithm compares all the possible alignments to enhance the chance to search for the best possible global alignment. The algorithm scans a pair of the given sequence for similarity and returns a pair of aligned sequences with the highest score. Let us consider two sequences S1 and S2 with lengths N and M as S1 = {s11,s12,s13, …, s1N} and S2 = {s21,s22,s23,…S2M}. The algorithm scans S1 and S2 from the left to the right by pairing nucleotides such as that it aligns each index of S1[i = 0,…,N] with the same index of S2[j = 0, …, M]. This alignment is called shift = 0. First, it scans, shift = 0 for these sequences, compares and finds the match score and a similarity matrix is maintained to hold the alignment position, matching score and shifting details. The similarity matrix can be imagined as a universal cluster (U). Thereafter, the sequence pair is shifted to the right or to the left (figure 2). The comparison process is repeated for all right (0→N) and left (0→−N) shifts. An evolutionary clustering algorithm[12] was used in the present manuscript to evolve the universal cluster (U) by considering the match score as a clustering factor that results in the evolution of many clusters with a different range of match scores. The final similarity matrix can be imagined as a set of clusters for all possible alignments of the sequences S1 and S2 with all possible shifts. During the process, paired alignments can migrate from one cluster to other as per their match score. It is made possible by the migration operator of the evolutionary algorithm[12]. For example, the clusters developed from the universal cluster can be considered as C1 = {86,98,67} and C2 = {27, 34,40} and their paired alignment with the match score of 86, 98, 67 and 27, 34, 40 respectively. A sort_cluster() method was applied in the present work to sort the evolved clusters in ascending order as per their match score. Therefore, the clusters were sorted as {C1, C2} because C1 consists of the paired alignment with the highest match scores. The top cluster (with the highest match score) was selected and marked as the final cluster. In the above example, C1 is the top cluster that is finally selected to find the best alignment. Sort_within_cluster() method was also applied to sort other alignments in the final cluster in ascending order that provides the best alignment on the top of other alignments in the final cluster. This is applied on the final cluster C1 that sorts the elements of C1 in an ascending order C1 = {98, 86, 67}. It is clear here that the paired alignment with the best score is C1 with 98 as the best match score. In this way the proposed algorithm not only provides the best possible pair-wise sequence alignment

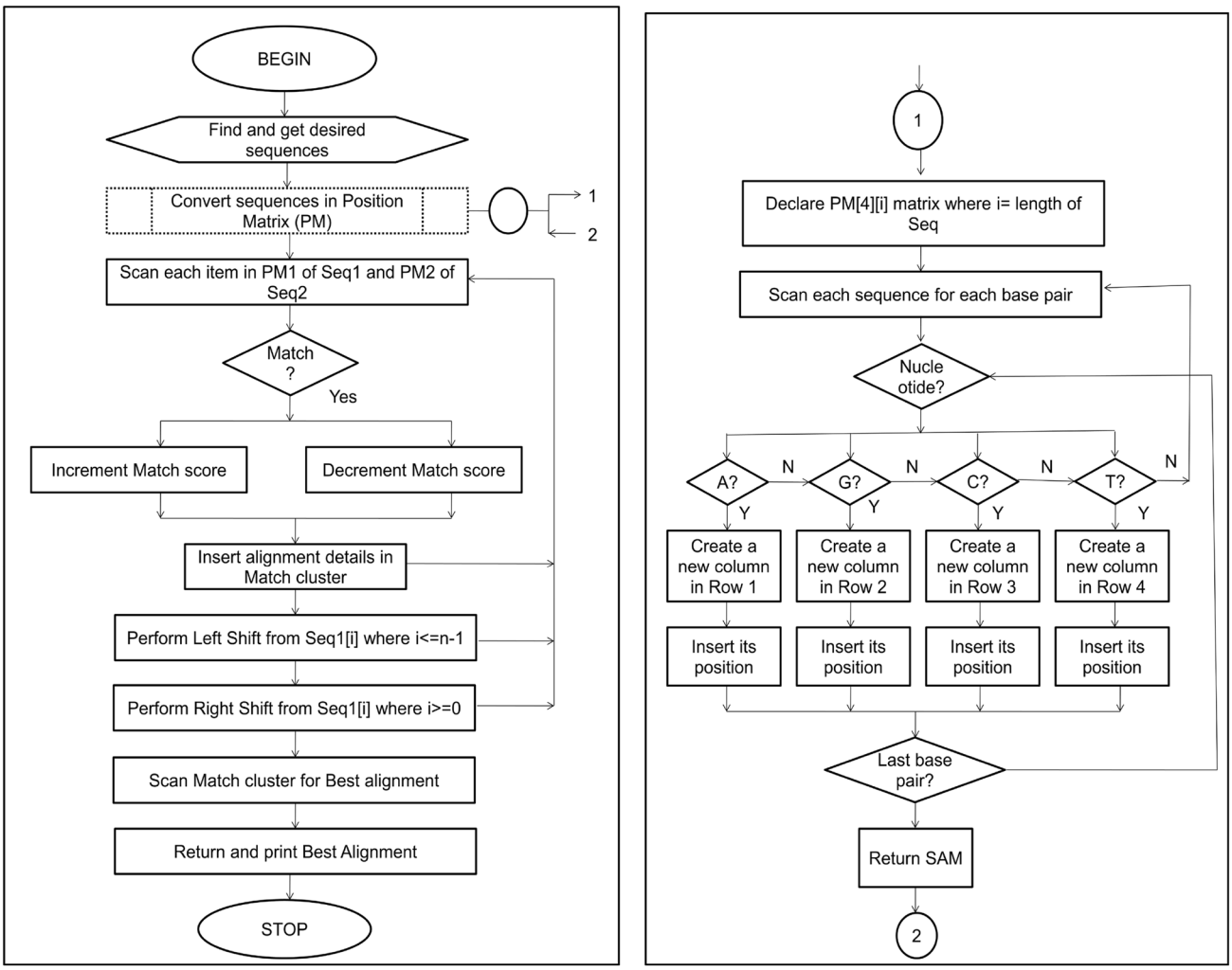

Figure 1. Flowchart for MPSAGA algorithm.

but also assist in finding out the remaining best possible alignments (with low match score) in ascending order.

2.2 *Description*

The proposed algorithm has been divided into five steps. The first step is the self-initialization step that initializes the parameters required to execute the algorithm. Then the algorithm scans each nucleotide of the sequence and converts the entire sequence in the Positional Matrix (PM). Positional Matrix is the novel representation of sequences and is being used in the present work for the first time. In this type of representation, in spite of storing a string of nucleotide, the position of each nucleotide is determined and located in a positional matrix accordingly. In the step 2, the possible alignment of the two sequences without gap and without shift is achieved. As a result, a universal cluster is generated with all the best possible pair-wise sequence alignments. Now the cluster is searched for shifting positions without inserting gaps. The process of conversion of a sequence in the positional matrix is given in the step 1 below and is described in detail in the following section. The positional matrix of S1 sequence is shown in figure 3.

The MPSAGA performs base pair match for all positive and negative shifts in steps 2 and 3. The details of the process of shifting, sequence alignment and assessment of sequence for insertions and deletions in the positional matrix representation model have been provided in the section 2. In step 4, the universal cluster is scanned using evolutionary algorithm ACERAM [12] to search for the best match score. The step 4 returns the best pair-wise sequence alignment with the highest match score and the last step 5 displays the finalized sequence alignment. The proposed algorithm has been detailed out below.

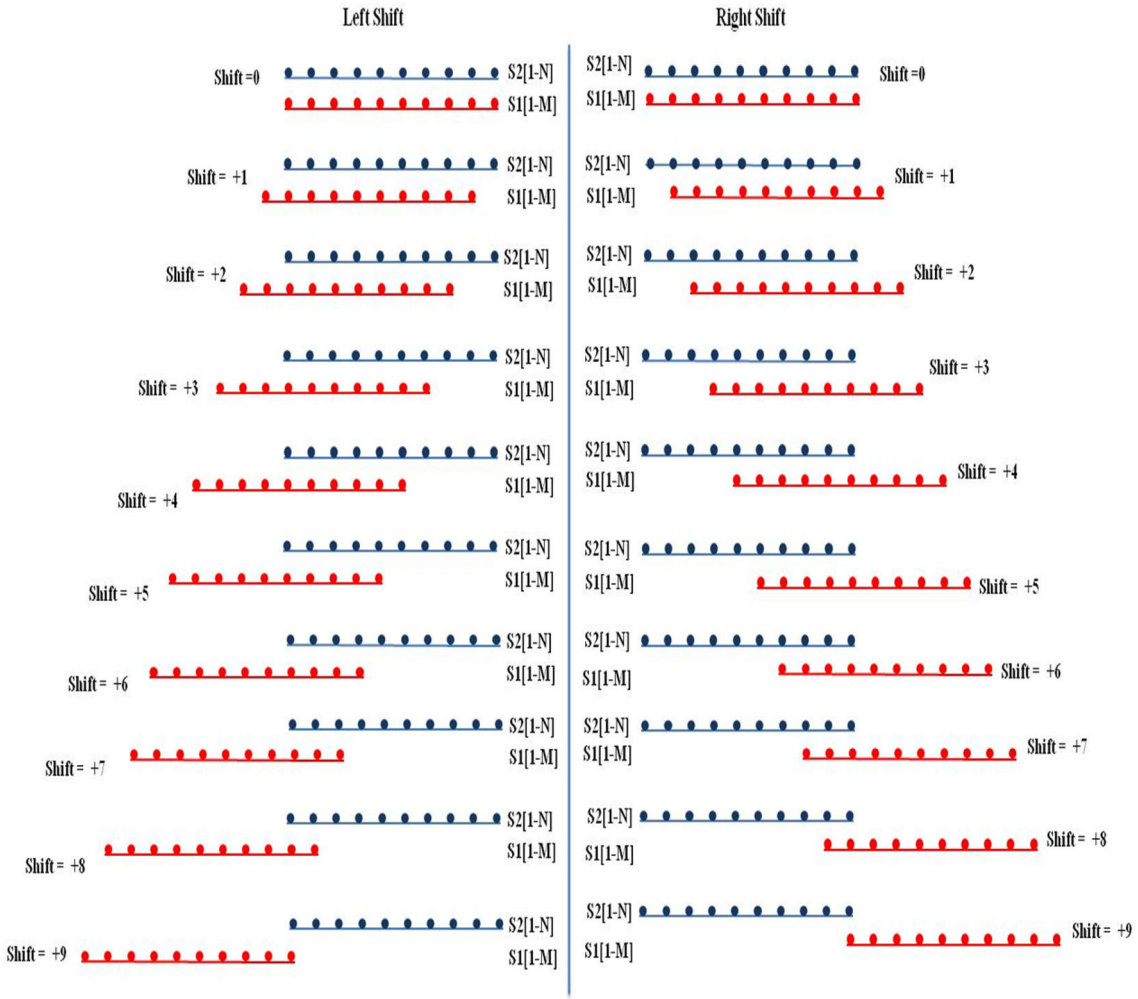

Figure 2. The process of shifting during pair-wise sequence alignment by MPSAGA. The process starts with aligning sequences without any shift (shift = 0) and the match-score is calculated. Thereafter, other possible alignments are generated by the proposed algorithm by shifting base pair in sequence1 one by one until the last base pair. This shifting can be performed from left (shift = −1) or right (shift = +1) which give rises to a new possible alignment. In each case the match score of the new possible alignment is calculated. All possible alignments of a sequence pair due to shifting process have been shown.

2.3 *Matrix based pair-wise sequence alignment algorithm for global alignment (MPSAGA)*

Input

Two Sequences S1 and S2 to be aligned of M and N lengths, respectively

Parameters

Range of Shift Parameter – (−N to +N)
Match Reward = +1
Mismatch penalty = − 1, if the substitution is of purine to purine (A→G or G→A transition) or pyrimidine to pyrimidine (C→ T or T→C transition) and mismatch penalty will be = − 2, if the substitution is purine to pyrimidine (A→ C, A→T, G→C, or G→ T transversion) or pyrimidine to purine (C→A, C→G, T→A or T→G transversion). One can use predefined similarity matrix like dnaall, PAM or BLOSUM for deciding match reward and mismatch panelty.

Output

An optimal sequence alignment and the formation of universal cluster for the population of auto evolved pair-wise sequence alignments as per their match scores (optional).

Step 1 [Initializes positional Matrix (PMs) for S1 and S2]
Two sequences are given here:
S1 = {s11, s12 .., s1M}
S2 = {s21, s22 .., s2N}
Declare PM1 and PM2 for S1 and S2. PM1 and PM2 are dynamic clusters represented as a matrix having four rows of the A, G, C and T nucleotides with dynamic number of columns. Positions of nucleotides in the sequence will be stored in PM. PM1 and PM2 are represented in figure 4.

A	A	G	C	C	T	T	A	G	C
0	1	2	3	4	5	6	7	8	9

A	0	1	7
G	2	8	
C	3	4	9
T	5	6	

Figure 3. Positional Matrix (PM) representation of a biological sequence S1.

Step 1.1 [positional matrix (PM) based sequence representation of S1 and S2]

```
Scan S1 to initialize PM1
  For i= 0 to M
      Read S1[i]
      If S1[i] = = 'A' then
          PM1[0]=i
      End if
      If S1[i] = = 'G' then
          PM1[1]=i
      End If
      If S1[i] = = 'C' then
          PM1[2]=i
      End if
      If S1[i] = = 'T' then
          PM1[3]=i
      End If
```

Repeat this same process to initialize PM2.

Step 2 [Find match score for alignment of PM1 and PM2 without gap and without shift]

Step 2.1 For each row(i) in PM1 and PM2
Step 2.2 For each column(j) in PM1 and PM2
Match the values of PM1[i][j] and If there exists a match Match_count++
Step 2.3 Find the total Match_count of each row
Step 2.4 Find the total Match_count for the complete alignment by adding Match_count of each row
Step 2.5 Insert the entries in Match_index matrix [Match_index matrix is the similarity matrix for universal cluster storing all possible pair-wise alignments]

Step 3 [Find Match score for alignment of PM1 and PM2 with shifts]

Step 3.1 Find the possible alignments for shift parameter from 1 to n−2 (positive shift) and −1 to −n−2 (negative shift)

For each value of shift parameter
For each value in PM2
Increment each value of PM2 by shift parameter

Step 3.2 Perform match by following all sub steps given in step 2 and insert it in Match_index matrix

Step 4 [Find best alignment of sequences S1 and S2]

To find the best alignment, scan and evolve the universal cluster and find an alignment with maximum Match_score

Step 4.1 [Calculate final score of possible alignments]
For all rows(i) in Match_index
Calculate Final_Match_score = Match_score -Miss_Matches
Insert Final_Match_score in Match_index[i][0]
Step 4.2 [Find best alignment by comparing Match_scores]
Call sort_within_cluster(sort_clusters())

Step 5 Print/Return best alignment
End of the Algorithm

2.4 *Space and time complexities*

The Needleman–Wunsch algorithm aligns two biological sequences of sizes n and m in a matrix of n x m whereas the proposed algorithm MPSAGA stores sequences in a positional matrix format. For biological sequences like DNA or RNA, the number of rows in a positional matrix is fixed as 4 and for the protein sequences, based on the number of amino acids, it is fixed as 21. Therefore, the space complexity of a DNA or RNA sequence can be expressed as 4 x maximum frequency of occurrence of one nucleotide. As expected, which is much less than the space complexity in Needleman–Wunsch algorithm. Thus, the proposed positional matrix sequence representation method of MPSAGA algorithm is useful in performing sequence comparisons and alignment tasks in a relatively shorter time than the conventional string method. The empirical study performed in the present work to compare the string method and the proposed PM method for sequence representation also suggests that up to ∼84% execution time can be saved in the pair-wise sequence alignment using the PM method. The time complexity of a sequence alignment using the string method is expressed as $O(n^2)$ whereas in the proposed PM method, it is expressed as O(min(m,n)*(n-shift)). The advantage of using conventional string representation method is its simplicity and its nature of representing biological sequences as a continuous string of monomers but it is important to note here that the process of pair-wise alignment of longer sequences using this method is very

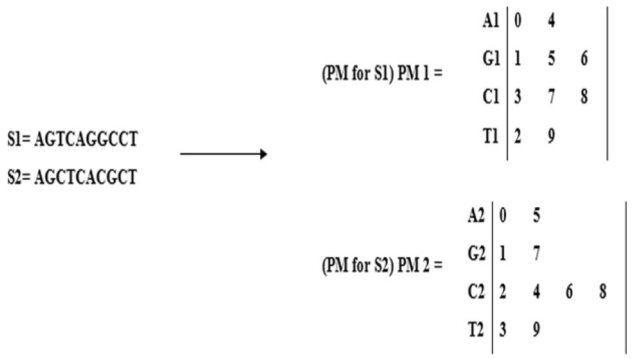

Figure 4. Positional Matrix (PM) representation of a biological sequence S1 and S2.

complicated that leads to extended execution time. Whereas, the proposed PM method converts each nucleotide in a pair of biological sequence in a numerical matrix (positional matrix) first which is relatively simpler and less bulky for a computational system to analyze and execute. Moreover, the PM method reduces effectively the time complexity of pair-wise sequence alignment of long and complex biological sequences.

3. Positional matrix representation model

In positional matrix representation, a sequence is represented as a special matrix called Positional Matrix (PM). Positional matrix contains the number of rows as per the number of building blocks (nucleotides/amino acids) in the sequence such as 4 rows of DNA for storing positions of 4 nucleotides (A,G,C,T), 4 rows for (A,G,C,U) in RNA sequence and 20 rows for protein sequence (for each amino acid). One additional row could be allotted for gapped alignments. Figure 3 is a positional matrix (PM) representation of a sequence S1. It is clear that nucleotide A is present at three positions (0, 1 and 7), G is present at two positions (2 and 8), C is present at three positions (3, 4 and 9) and T is present at two positions (5 and 6). Thus, PM based sequence representation reduces the space complexity of the proposed MPSAGA.

3.1 *Procedure to use PM representation for sequence alignment*

The proposed algorithm uses PM to represent sequences and scans the two sequences under investigation at once in the complete algorithm.

The two sequences are

S1 = AGTCAGGCCT

S2 = AGCTCACGCT

The sequence alignment matrix for S1 and S2 are PM1 and PM2 respectively, and are presented in figure 4.

3.2 *To calculate matching score using PM*

The matching score is calculated by scanning the PM indexes. If the PM1 and PM2 values at any index are same then it is a match. In the above example, there are four matches at A1[0] and A2[0], G1[0] and G2[1], C1[2] and C2[4] and T1[1] and T2[1]. Hence the matching score is 4.

3.3 *To implement positive shifting in PM*

To implement positive shifting (right shift) in the alignment, one has to just add shifting index in each value of PM2 from 0 to N. For example, the +1 shift in S2 can be calculated as shown in figure 5. S2 sequence has only 0–9 indexes therefore T2[1] = 10 is limit out and can be eliminated in overlapped sequence alignments from the further processing. The complete process is shown in figure 6.

3.4 *To implement gaps in PM*

If gap is considered in the alignments the sequence alignment of the positional matrix will be

S2 =_ AGCTCACGCT

And if, gap exists, then it will be a matter of gap penalty (Figure 7). Gaps are not permitted in the proposed algorithm, but represented here for the sake of clarity of the procedure.

3.5 *To implement insertions and deletions in PM*

Insertions and deletions by point mutations and exchange of nucleotides in DNA sequences are common in nature. To implement insertions and deletions (in-del) the proposed method accepts maximum two insertions or deletions. Each base in the first sequences is scanned and checked for its match in the opposite, second sequence with −2, −1, 1, +1 or +2 possibilities. If there is no match, then the insertion and deletion is possible. Figure 8a shows a perfect base pair match at position 0 between S1 and S2. Therefore, no shift or in-del is required and the match score will remain 4. So no change will be made in PM1 and PM2. At position 1, G nucleotide in both S1 and S2 is a direct match; therefore no in-del is required here at this position. PM1 and PM2 will

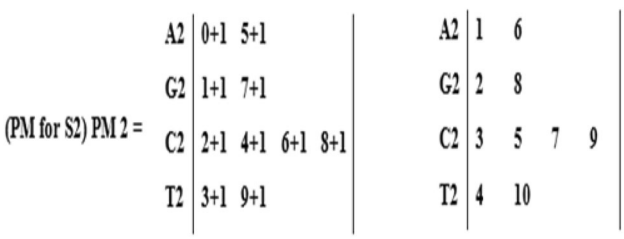

Figure 5. A positive shift in sequence S2.

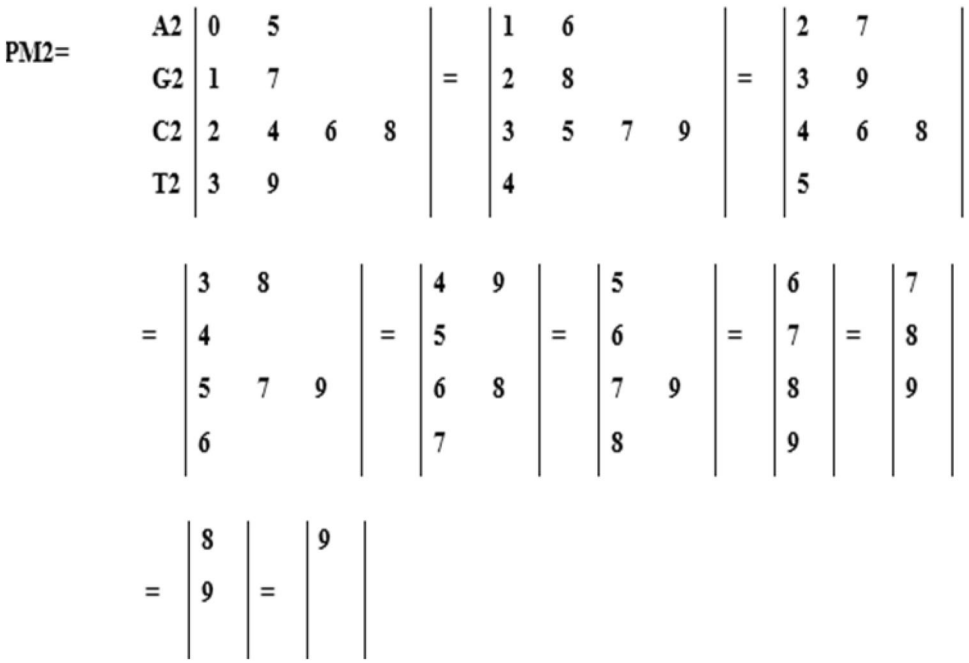

Figure 6. Positive shifting in position based matrix from index 0 to 9.

remain the same at this point. The match score will remain 4 in this position (figure 8(b)). At position 2 of S1 and S2, there is no direct base pair match. Nucleotide T at position 3 in S2 can be matched with nucleotide T at position 2 in S1 and hence there is an exchange. The process of exchange of nucleotide T in position matrixes of sequences S1 and S2 is shown in figure 8c. After this step there are total 6 base pair matches in S1 and S2 sequences. As the exchange operator is used for this step, the final match score will be the match score – cost of exchange operator (2 i.e., a default value). Therefore, the final match score will be 4 after this step. There is a direct match again at position 3, therefore no change is required and the match score will remain 4 (figure 8(d)). The process of base pair alignment at position 4 is shown in figure 8e. Because there is no direct match at this position and an exchange will occur from C to A, therefore the match score will become 7. The exchange penalty is −2, hence the match score will be 5. In this way, the MPSAGA will process all the base pairs till position 9 to achieve the best possible alignment between S1 and S2. The final pair of aligned sequences is shown in figure 8f with all possible 9 base pair matches between the two sequences.

4. Empirical study

4.1 *Data collection*

Forty two pairs of sequences were randomly selected and downloaded from NCBI (National Center for Biotechnology Information) nucleotide database. These pairs of sequences vary in length (table 1).

4.2 *Experimental set-up*

The experiment was performed on Pentium® Dual Core CPU T4300 @ 2.10 GHz, 4 GB RAM and Windows 7, 64 bit platform. The proposed algorithm was implemented and executed using Java 8 on Eclipse Mars IDE. The algorithm, MPSAGA is available in public domain at https://github.com/JyotiLakhani1/MPSAGA and the snapshot of its execution has been provided in supplementary material as Appendix A. The classical pair-wise algorithm, Needleman–Wunsch algorithm was treated as a control and has been implemented on eclipse using Java8 with plug-in of

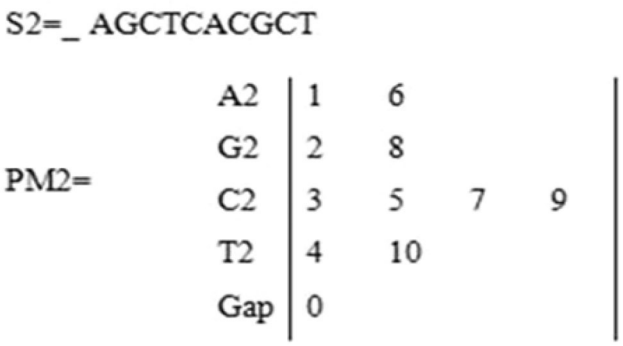

Figure 7. Positional Matrix (PM) representation of sequence 2 with gaps.

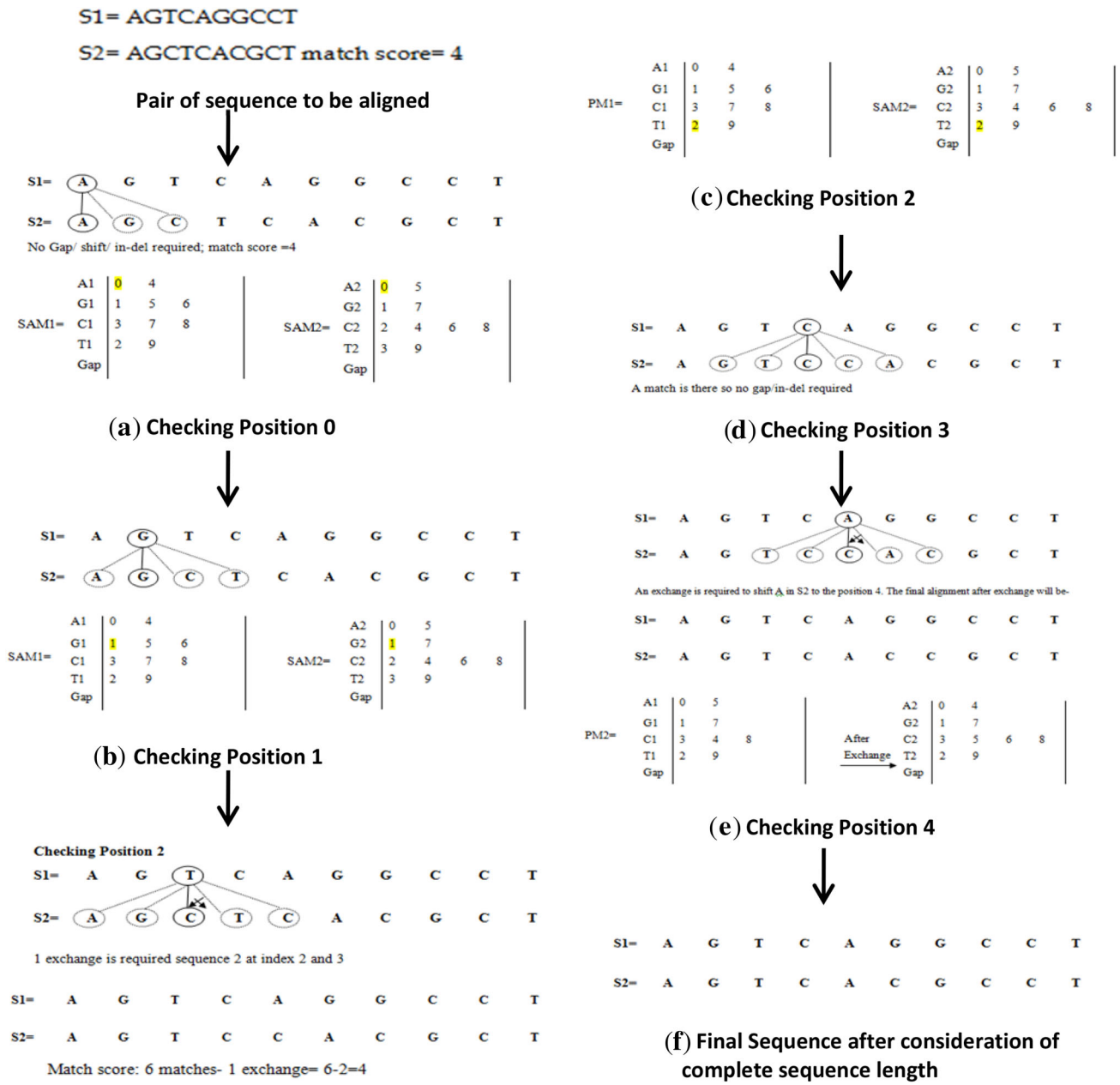

Figure 8. The process of pair-wise sequence alignment by MPSAGA using positional matrix representation. S1 and S2 sequences have been shown in string as well as in positional matrix representation. The sequence alignment is executed by MPSAGA by comparing each base of S1 from left to right to the bases in S2 located opposite to it. If match is not found, it performs insertion or deletion or exchange operation for the best possible sequence alignment. The match score is 4 initially (4 base pair match). (**a**) At first, the position i in S1 (that is base A) is compared with i+2 to i−2 positions in S2. i−1 and i−2 positions do not exist this time. There is a direct match at i position for base A in S1 and S2. No comparison for this position will be performed and the match score will remain 4. (**b**) The second base G (position 1) in S1 is compared with A, G, C, T bases at i−1, i, i+1, i+2. There is also a direct match at position 1 (G to G), therefore, the match score will remain 4. (**c**) The base T at position 2 in S1 is compared with S2 for the same position and there is no direct match and hence MPSAGA process positions i−2. i−1, i, i+1, i+2 for sequence alignment. There is a match at i+1 position, hence C in S2 is exchanged with T in S2. The match score will turn 6 but there will be an exchange penalty of −2, consequently the match score will remain 4. (**d**) There is a direct match at position 3. (**e**) At position 4, base A in S1 is compared with S2 for the same position. Since there is no direct match again there will an exchange at position i+1 from C to A and the match score becomes 7. The exchange penalty is −2, hence the match score will reduce to 5 and so on *(f)* MPSAGA will process all the base pairs till position 9 for the best possible alignment between S1 and S2, the final pairwise sequence alignment of S1 and S2 has been provided.

Table 1. NCBI sequence IDs and the size of forty two pairs of sequences.

Sl. No.	Sequence1	Size of Seq1	Sequence2	Size of Seq2
P1	FR850916.1	181	FR850691.1	153
P2	FN432372.1	1050	FN432373.1	1034
P3	L06042.1	9597	NC_001802.1	9181
P4	JX650235.1	600	JX650236.1	602
P5	Y00277.1	9646	M80208.1	1461
P6	X57323.1	808	X06879.1	1142
P7	X57323.1	808	L06042.1	9597
P8	X57323.1	808	X07805.1	9170
P9	X57323.1	808	M80208.1	1461
P10	X57323.1	808	Y00277.1	9646
P11	X57323.1	808	X07805.1	9170
P12	X57323.1	808	M80208.1	1461
P13	X06879.1	1142	L06042.1	9597
P14	X06879.1	1142	Y00277.1	9646
P15	X06879.1	1142	M80208.1	1461
P16	X06879.1	1142	X07805.1	9170
P18	L06042.1	9597	M80208.1	1461
P19	L06042.1	9597	Y00277.1	9646
P21	Y00277.1	9646	M80208.1	1461
P22	X07805.1	9170	M80208.1	1461
P24	M80208.1	1461	Y00277.1	9646
P25	M80208.1	1461	X07805.1	9170
P26	I03176.1	99	I03178.1	102
P27	KC506162.1	2384	KC506163.1	2246
P28	JX985678.1	366	JX985681.1	345
P29	HQ711856.1	339	HQ711857.1	339
P30	HM856336.1	732	HM856337.1	732
P31	KU708253.1	2381	KU708254.1	2200
P32	KU159362.1	6840	KU159363.1	7030
P33	KU159363.1	7030	KU159357.1	7271
P34	EF591314.1	1575	EF591316.1	1572
P35	JX129454.1	1009	JX129455.1	1027
P36	JX129464.1	906	JX129465.1	907
P37	KM206982.1	1260	KM206949.1	1257
P38	U10689.1	4741	U10690.1	4736
P39	L31857.1	1451	L31858.1	1443
P40	L11083.1	1746	K00490.1	1696
P41	FJ716745.1	1521	AB762357.1	1630
P42	GQ892202.1	1049	GQ892203.1	1048

biojava4.0.0 with the help of maven and ant tools. The proposed algorithm has also been compared with other global pair-wise alignment tools. Forty two pairs of sequences were converted to the proposed PM based format for data representation.

4.3 *Parameter set-up*

Following parameters were used to perform the experiments in the present study to implement MPSAGA, The parameters and their default values are furnished in the discussion part. The Range of shift parameter = −N to +N, Shifting = true, gap = 0, Match Reward = +1, Mismatch Penalty = −1 for transitions (substitution of Purine > Purine (A to G or G to A) or Pyrimidine→Pyrimidine (C to T or T to C)), Mismatch Penalty = −2 for transversions (substitutions of Purine to Pyrimidine (A to C, A to T, G to C or G to T) or Pyrimidine to Purine (C to A, T to A, C to G or T to G)). Here A, G, C, and T are Adenine, Guanine, Cytosine, and Guanine, respectively. Other parameters are Insertion deletion penalty = −1, Threshold = no limit by default, Distance measure= Hamming distance. The Hamming score has been used to find out the match score where only matches and mismatches are considered.

Table 2. Comparison of time complexity of three algorithms for a pair-wise global alignment of two nucleotide sequences of *Rhizobium leguminosarum* partial *nifh* genes for nitrogenase fe protein.

Algorithms	Parameters Used	No. of Match	No. of Mismatch	No. of Gap	Shift	Shift Offset	Match Score	Execution Time
MPSAGA	match = +1 mismatch = −1 gap = no gap Match Score = Hamming Score	137	49	–	+1	36	88	24 milliseconds
nBLAST	match = +1 mismatch = −1 gap = no gap	137	13	36	+1	0	81	–
Needleman–Wunsch	–	137	–	–	+1	36	–	340 milliseconds

5. Complexity analyses

Suppose S1 and S2 are sequences of size m and n respectively. S1 is the query sequence which is to be aligned with the sequence S2. Initially, the sequence pair S1 and S2 will be aligned by MPSAGA aligner without any shift, therefore the complexity of alignment for this step will be min (m,n). In the next step, the algorithm scans S1 and S2 from the left to the right by shifting and pairing nucleotides in the sequence pair. At this point, the complexity will be dependent on the actual size of sequence alignment which is -

Size of S2-shift parameter = (n- no of shifts)

So the complexity of this algorithm will be min(m,n)*(n-shift).

For example-

Without shift-

0123456789 (index)

S1: ********** (size = m = 10)

S2: ******** (size = n = 8)

Alignment length = min(m,n) = 8

In further steps-

After 3 negative shifts-

0123456789 (index)

S1: ********** (size = m = 10)

S2:******** (size = n = 8)

Alignment length = n-shift = 8−3 = 5

6. Results

The experiments performed in the present research had three main objectives. The first objective was to test the effectiveness of the proposed algorithm. The second objective was to perform time and space complexity analysis using MPSAGA and other algorithms and the third objective was to test the Position Matrix based representation model. To address the first one, Needleman–Wunsch algorithm, nBLAST [28–30] and the proposed MPSAGA were compared to find out the best global pair-wise alignment algorithm with respect to the execution time. nBlast algorithm is a popular tool to achieve the optimal results. We have used nBLAST (online tool) with parameters setting such as match = +1, mismatch = −1 to get the optimal alignment. We have discussed this earlier that the online tools work faster than the local tools because they are located on the high configuration servers. Three other global pair-wise algorithms that are Emboss-Needle, ALIGN and LALIGN were also tested for 10 randomly selected sequence pairs (table 1). The result indicated that the MPSAGA executed pair-wise sequence alignment much faster with no gaps than the Needleman–Wunsch algorithm and nBLAST. The result of one such execution is being described here. A pair-wise global alignment of two nucleotide sequences of *Rhizobium leguminosarum* partial *nifH* gene of nitrogenase Fe protein is given in table 2 (Sequence 1 ID = FR850916.1 for strain R46098 (181 base pair long) and Sequence2 ID- FR850691.1 for strain R45920 (153 base pair long)). On execution of three different algorithms for S1 and S2 pair-wise sequence alignment, MPSAGA effectively outcompeted Needleman–Wunsch and noblest algorithms (please see table 2). The nBLAST also yielded the equivalent result, but with 36 number of gaps from location 2 to 37. Thus, it can be stated that as compared to Needleman–Wunsch algorithm (340 ms) and nBLAST, the best sequence alignment in minimum execution time can be achieved using MPSAGA (24 ms).

It is inappropriate to calculate the time taken by nBLAST because the nBLAST is an online tool and the time taken by nBLAST to align two sequences may also include delays due to the network and server connectivity. To find out the significance level of MPSAGA relative to Needleman-Wunsch, nBLAST, Emboss-Needle, ALIGN and a LALIGN algorithm, a normal distribution of their z score has been provided that clearly underlines MPSAGA for its statistically fit identity (figure 9). The atypical result with nBLAST may be attributed to the un-gapped cases considered in the present experiment while nBLAST is created

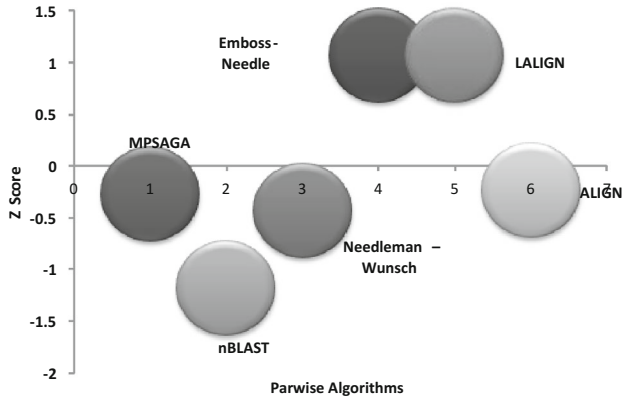

Figure 9. Distribution of z-scores for MPSAGA and four other global pair-wise sequence alignment algorithms.

to provide alignment of sequences having gaps. The present results of sequence alignment using the above algorithms confirm that the best possible pair-wise sequence alignment is achieved in a relatively small time using MPSAGA. These results clearly indicate that the proposed algorithm is capable of finding the best sequence alignment as same as the most popular algorithm Needleman–Wunsch algorithm finds. To address the second objective to compare the space and time complexities of the proposed algorithm, 45 runs of MPSAGA and NW- Blast algorithms were executed for the paired sequences listed in table 1. The output has been given in figure 10. The majority of times MPSAGA outperformed than the Needleman–Wunsch algorithm, but for a few of the sequence pairs, the performance of both of the algorithms was same. It is important to note that good algorithms execute pair-wise sequence alignment with minimum space and time complexities. To comprehend these results more fully, we performed another evaluation of these two algorithms in relation to the size of paired sequences. We also tried to find out the relation between the number of identical pairs and the number of gaps inserted in the sequence for effective pairing by NW-Blast algorithm. To perform this evaluation, three clusters of sequences were created according to the size of Sequence 1 (figure 11). The cluster 1 represents sequences with 0<size<1000 base pairs. The cluster 2 represents sequences with 1000<size<5000 base pairs and finally the cluster 3 represents sequences with 5000<size<10000 base pairs. To mention here, MPSAGA is a un-gapped algorithm, whereas NW-Blast (Needleman–Wunsch Blast) algorithm include gaps in a sequence when aligning a pair of sequence.

Figure 11a clearly indicates that highest match (identity) for a pair of similar sized sequences was observed with MPSAGA followed by NW-Blast algorithm. After P28 sequence pair (figure 11(a)), on x axis, the performance of MPSAGA is declining. The reason for this observation can be attributed to the difference of size between the two sequences and the incorporation of gaps by NW-Blast algorithm that has a clear impact on the output (number of matches) (figure 11(b)). It is noteworthy that NW-Blast is inserting gaps to justify the unmatched alignment and size differences otherwise the identical matches in NW-Blast and MPSAGA algorithms are comparable. For middle sized sequences, there was no much difference in the output for the two algorithms (figure 11(c) and (d)). A very dramatic relationship was seen in the long sequences (figure 11(e) and (f)) It is highly likely that gaps are inserted here by NW-Blast for effective alignment. But the results clearly revealed that with increase in sequence size, MPSAGA still performs nicely (11(e)). However, the complexity increases with the inserted gaps because MPSAGA is restricted to perform sequence alignments having no gaps (11(f)).

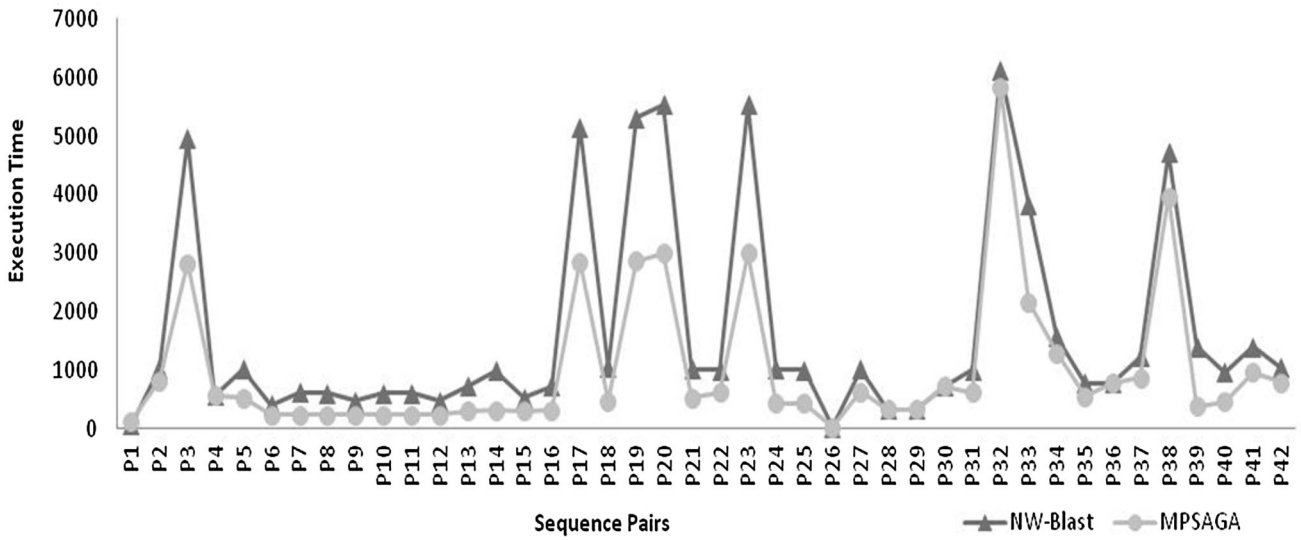

Figure 10. A direct comparison of Needleman–Wunsch and MPSAGA algorithms.

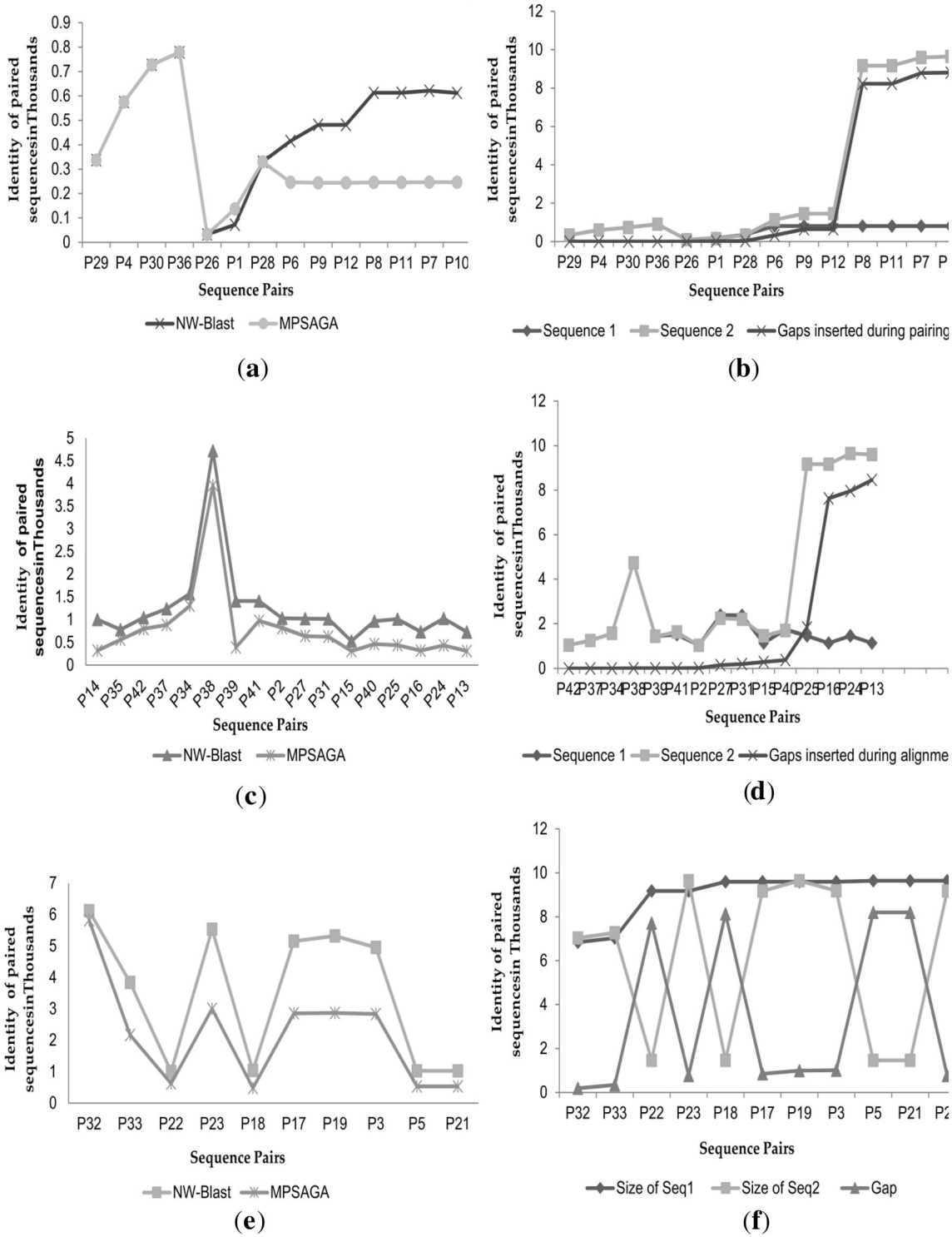

Figure 11. Comparison of outputs of Needleman–Wunsch and MPSAGA algorithms. **(a)** Sequence1 with size <1000 bp. **(b)** With relation to inserted gaps in in sequence1 with size <1000 bp. **(c)** Sequence1 with 5000 bp>size <1000bp. **(d)** With relation to gap in sequence1 with 5000 bp < size <=1000 bp. **(e)** Sequence1 with 10000<=size>5000 bp. **(f)** With relation to gap in sequence1 with 10000<=size>5000 bp.

In a separate experiment, additional 932 pairs of random nucleotide sequences with varied length and similarity were pair-wise aligned using MPSAGA with the same parameters as these were used in the above mentioned analysis. The detail of these 932 nucleotide sequences has been provided in supplementary material as Appendix B.

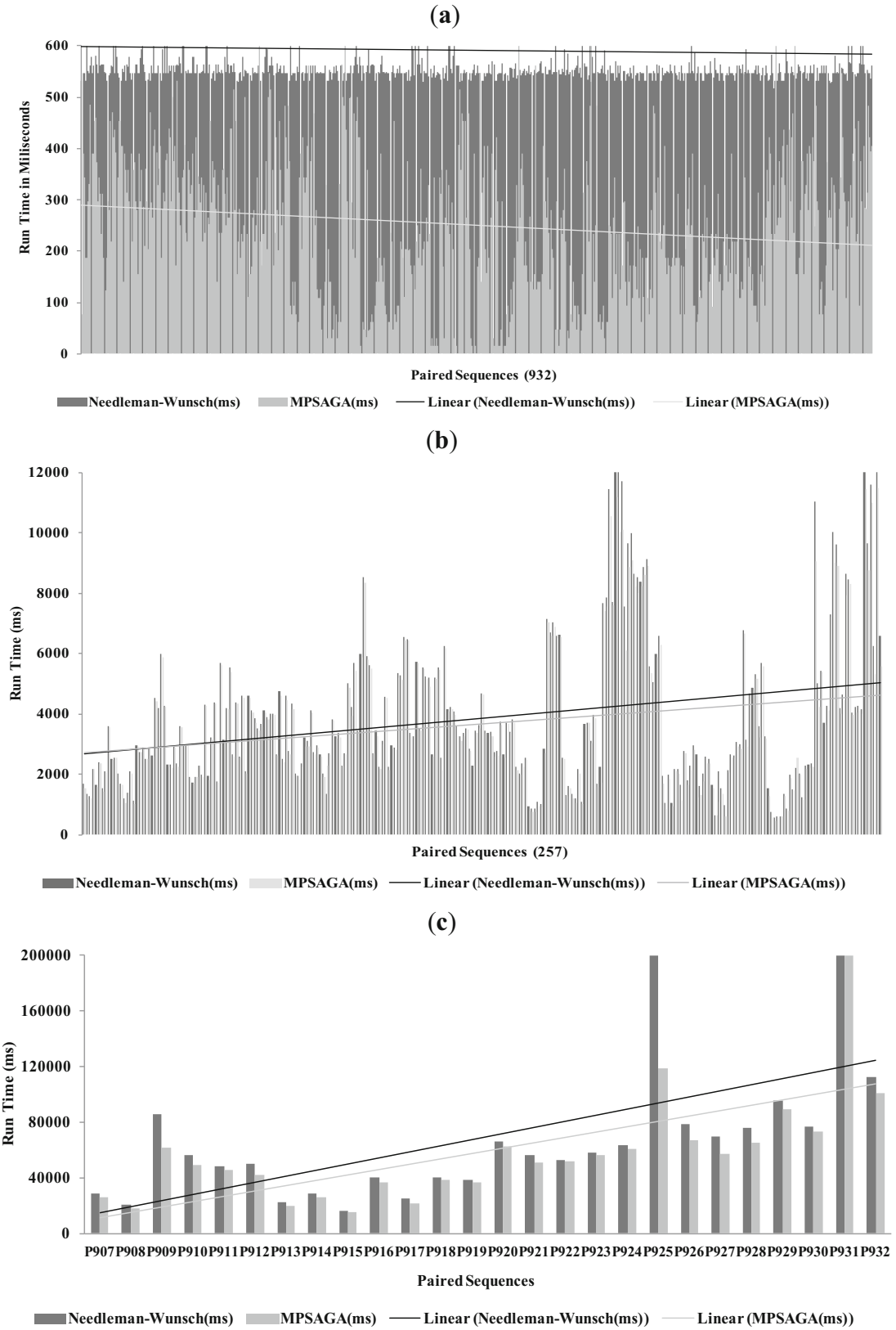

Figure 12. Comparison of execution time using MPSAGA for pairwise alignment of random 932 biological sequences. **(a)** 649 sequences with size 0<size<1000 bp. **(b)** 257 sequences with size 1000<size<5000 bp. **(c)** 26 sequences with size 5000<size<10000 bp.

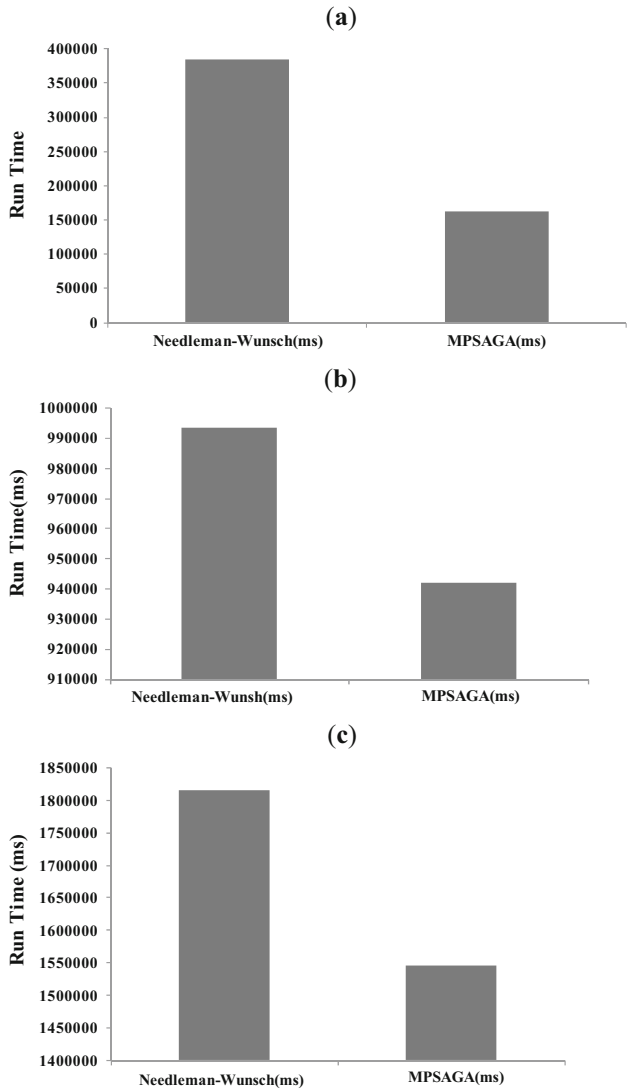

Figure 13. Average run time taken by Needleman–Wunsch and MPSAGA algorithims for aligning sequence pairs. (**a**) Sequences with size 0<size<1000 bp. (**b**) Sequences with size 1000<size<5000 bp. (**c**) Sequences with size 5000<size<10000 bp.

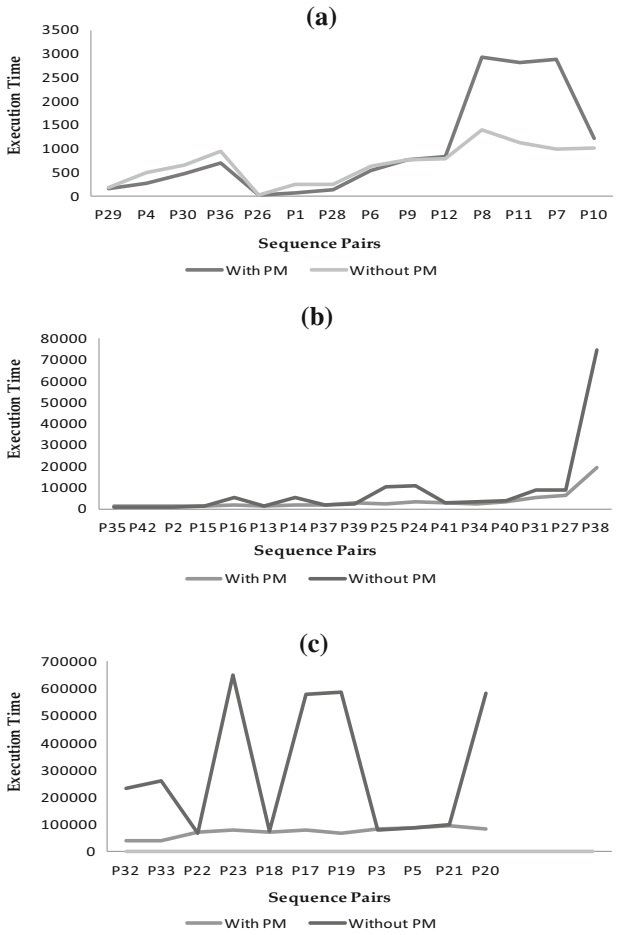

Figure 14. Comparison of execution time using MPSAGA with and without Position Matrix Representation. **(a)** Sequences with size 0<size<1000 bp. **(b)** Sequences with size 1000<size<5000 bp. **(c)** Sequences with size 5000<size<10000 bp.

All 932 pairs of nucleotide sequences were clustered on the basis of the size of sequence 1 where the cluster 1 represents 649 sequences with 0<size<1000 base pairs. The cluster 2 represents 257 sequences with 1000<size<5000 base pairs and the cluster 3 represents 26 sequences with 5000<size<10000 base pairs. The results from the analysis have been presented in figures 12a–c. It is clear from this observation too that performing pair-wise sequence alignment using MPSAGA is relatively faster than the Needleman–Wunsch algorithm. The pivot analysis of the three different clusters of nucleotide sequences and their average run time taken in executing pair-wise sequence alignment also establishes the dominance of MPSAGA over Needleman–Wunsch algorithm (13(a–c)).

To address the final objective, that is to see the effectiveness of the proposed novel PM representation, sequence alignment was represented as string and PM using MPSAGA. The experiment was implemented in Java on Mars version of the Eclipse platform. Forty two pairs of sequences (table 1) were used to perform this experiment. Execution time of MPSAGA with PM as well as with the conventional string method of sequence representation was compared for three clusters of sequences where the cluster 1 represents sequences with 0<size<1000 base pairs. The cluster 2 represents the sequence with 1000<size<5000 base pairs and finally the cluster 3 represents the sequence with 5000<size<10000 base pairs. The output of this experiment is shown for these three clusters in figures 14a–c. The algorithm, MPSAGA when executed with PM takes more time to analyze small sized sequences of cluster 1 (figure 14(a)). The reason for this observation may be attributed to the high overhead cost of converting cluster 1 (pairs of short sequences) to PM as compared to processing the small strings. And it could be that the conversion of cluster 2 and 3 in PM and its associated overhead cost is compensated to the process of analyzing complex and long string sequences by MPSAGA. But as shown in figure 14(b) and (c), it is clear

that the performance of MPSAGA with PM based sequence representation of clusters 2 and 3 is much more superior to the string representation.

7. Conclusions

The proposed algorithm for pair-wise sequence alignment using a novel positional matrix representation represents a more committed way to analyze biological sequences. The blend of the proposed MPSAGA with position based representation is intrinsically simple to execute and is capable of reducing the space and time complexity issues of previous algorithms significantly. The algorithm, MPSAGA results in the fast pair-wise sequence alignment with the same number of matching base pairs comparable to Needleman–Wunsch and other algorithms. In addition, observations also strengthen the fact that MPSAGA is competent enough to find the best possible pair-wise sequence alignment results as same as observed using nBLAST algorithm. The algorithm, MPSAGA executed pair-wise sequence alignment in 18.348% less time when compared to Needleman–Wunsch algorithm in the same experiment. The best possible sequence alignment similar to Needleman–Wunsch algorithm for un-gapped pairing was observed using MPSAGA. The experiments pertaining to the utility of MPSAGA with and without PM based representation of sequences suggest that there is a strong positive correlation between the size of the sequences and the performance of MPSAGA. The evidence from concluding experiments also supports that MPSAGA with positional matrix sequence representation is the best approach to align small, middle and long-sized biological sequences and highly sustainable to reduce the time and space complexities associated with earlier algorithms. The proposed algorithm MPSAGA, when compared with nBLAST with the same number of matching pairs, except gaps, exhibited the best possible sequence alignment results. The proposed algorithm can be applied to biological sequence analysis problems where the exact match between two sequences is required, such as in gene searches and in finding protein motifs. The algorithm can also be used for natural language processing for phrase alignment, text summarization, etc. Similarly, the positional matrix representation may be used in cryptography for message encryption and decryption. The improvisation of the proposed algorithm in relation to the larger sequences with gaps is the subject of future study.

References

[1] Baker W, Broek A, Camon E, Hingamp P, Strek P, Stoesser G and Tuli M A 2000 The EMBL Nucleotide Sequence Database. *Nucleic Acid Res*. 28(1): 19–23

[2] Cochrane G, Aldebert P, Althorpe N, Andersson M, Baker W, Baldwin A, Bates K, Bhattacharyya S, Browne P, Van denBroek A *et al* 2006 EMBL Nucleotide Sequence Database: developments in 2005. *Nucleic Acids Res*. 34:10–15

[3] Benson D A, Karsch-Mizrachi I, Lipman D J, Ostell J and Sayers E W 2011 Genbank. *Nucleic Acids Res*. D39:32–37

[4] Benson D A, Karsch-Mizrachi I, Lipman D J, Ostell J and Wheeler D L 2007 GenBank. *Nucleic Acids Res*. 35:D21–D25

[5] Okubo K, Sugawara H, Gojobori T and Tateno Y 2006 DDBJ in preparation for overview of research activities behind data submissions. *Nucleic Acids Res*. 34:6–9

[6] Tateno Y, Imanishi T, Miyazaki S, Fukami-Kobayashi K, Saitou N, Sugawara H and Gojobori T 2002 DNA Data Bank of Japan (DDBJ) for genome scale research in life science. *Nucleic Acids Res*. 30(1):27–30

[7] Schuler G D, Epstein J A, Ohkawa H and Kans J A 1966 Entrez: molecular biology database and retrieval system. *Methods Enzymol*. 266:141–162

[8] Pearson W R and Lipman D J 1988 Improved Tools for Biological Sequence Comparison. *Proceedings of the National Academy of Sciences of the United States of America*, pp.2444–2448

[9] Altschul S F, Gish W, Miller W, Myers E W and Lipman D J 1990 Basic local alignment search tool. *J. Mol. Biol*. 215 (3): 403–410

[10] Kumar S, Tamura K and Nei M 1994 MEGA: Molecular Evolutionary Genetics Analysis software for microcomputers. *Comput. Appl. Biosci.* 10(2):189–191

[11] Kumar S, Stecher G and Tamura K 2016 MEGA7: Molecular Evolutionary Genetics Analysis version 7.0 for bigger datasets. *Mol. Biol. Evol*. 33(7):1870–1874

[12] Lakhani J, Khunteta A, Chowdhary A and Harwani D 2016 Auto-Evolving Clusters based on Rejection and Migration. In: *Proceedings of the International Conference on Advances in Information Communication Technology & Computing (AICTC '16)* Bishnoi S K, Kuri M, Goar V (Eds.). ACM, New York, NY, USA,, Article 98, 6 pages

[13] Smith T F and Waterman M S 1981 Identification of common molecular sub sequences. *J. Mol. Biol.* 147:195–197

[14] Needleman B and Wunsch D 1970 A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol*. 48(3): 443–53

[15] Chakraborty A and Bandyopadhyay S 2013 FOGSAA: fast optimal global sequence alignment algorithm. *Sci. Rep.* 3:1746

[16] Morgenstern B 2004 DIALIGN: Multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Res.* 32:W33–W36

[17] Kurtz S, Phillippy A, Delcher A L, Smoot M, Shumway M, Antonescu C and Salzberg S L 2004 Open source MUMmer 3.0, Versatile and open software for comparing large genomes. *Genome Biol.* 5: R12

[18] Delcher A L, Phillippy A, Carlton J and Salzberg S L 2002 MUMmer 2.1, NUCmer, Fast algorithms for large-scale genome alignment and comparision. *Nucleic Acids Res*. 30(11): 2478–2483

[19] Delcher A L, Kasif S, Fleischmann R D, Peterson J, White O and Salzberg S L 1999 MUMmer 1.0, Alignment of whole genomes. *Nucleic Acids Res*. 27(11): 2369–2376

[20] Weichun H, David M, Umbach and Leping Li 2006 Accurate anchoring alignment of divergent sequences. *Bioinformatics.* 22:29–34

[21] Wilkinson L 1999 Dot plots. *The American Statistician.* 53:276–281

[22] Stamm M, Staritzbichler R, Khafizov K and Forrest L R 2014 AlignMe-a membrane protein sequence alignment web server. *Nucleic Acids Res*.42(W1):W246–251

[23] Stamm M, Staritzbichler R, Khafizov K and Forrest L R 2013 Alignment of Helical Membrane Protein Sequences Using AlignMe. *PLoS ONE.* 8(3):e57731

[24] Khafizov K, Staritzbichler R, Stamm M and Forrest L R 2010 A study of the evolution of inverted-topology repeats from LeuT-fold transporters using AlignMe. *Biochemistry* 49(50):10702–10713

[25] Gentleman R, Carey V, Huber W, Irizarry R and Dudoit S 2005 *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer

[26] Gentleman R 2008 *Programming for Bioinformatics*. Chapman and Hall, CRC Press, Boca Raton

[27] Hahne F, Huber W, Gentleman R and Falcon S 2008 *Bioconductor Case Studies*. Springer

[28] Vogt N 2016 NBLAST: a similarity search for neurons. *Nature Methods* 13:717

[29] Dumontier M and Hogue C W V 2002 NBLAST: a cluster variant of BLAST for NxN comparisons. *BMC Bioinfo.* 3:13

[30] Smith K 2014 *A brief history of NCBI's formation and growth*. The NCBI Handbook. 2nd edition [Internet] ix-xiv