Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment

Farhana Naznin, *Member, IEEE,* Ruhul Sarker, *Member, IEEE,* and Daryl Essam

***Abstract*—In this paper, we have proposed a progressive alignment method using a genetic algorithm for multiple sequence alignment, named GAPAM. We have introduced two new mechanisms to generate an initial population: the first mechanism is to generate guide trees with randomly selected sequences and the second is shuffling the sequences inside such trees. Two different genetic operators have been implemented with GAPAM. To test the performance of our algorithm, we have compared it with existing well-known methods, such as PRRP, CLUSTALX, DIALIGN, HMMT, SB_PIMA, ML_PIMA, MULTALIGN, and PILEUP8, and also other methods, based on genetic algorithms (GA), such as SAGA, MSA-GA, and RBT-GA, by solving a number of benchmark datasets from BAliBase 2.0. To make a fairer comparison with the GA based algorithms such as MSA-GA and RBT-GA, we have performed further experiments covering all the datasets reported by those two algorithms. The experimental results showed that GAPAM achieved better solutions than the others for most of the cases, and also revealed that the overall performance of the proposed method outperformed the other methods mentioned above.**

***Index Terms*—Dynamic programming (DP), genetic algorithm (GA), guide tree, multiple sequence alignment (MSA), progressive alignment.**

I. Introduction

MULTIPLE sequence alignment (MSA), the simultaneous alignment among three or more nucleotide or amino acid sequences, is one of the most essential tools in molecular biology. Sequence alignments are used to help demonstrate homology between new and existing sequences, to suggest primers for polymerase chain reaction, and to predict the secondary or tertiary structure of RNA and proteins [1], [2]. Therefore, the development of efficient and accurate automatic methods for multiple sequence alignments is a very important research topic.

Sequence alignment is the arrangement of two or more sequences of "residues" that maximizes the similarities between them. In order for a multiple alignment to be meaningful in this context, all sequences in the multiple alignment must have a common origin. The goal of multiple sequence alignment is to align sequences according to their evolutionary relationships. MSA is important because it reconstructs phylogenetic trees, which in turn predict the function of an unknown protein by aligning its sequences with some other known functions. The various match, mismatch, and indel ("-") events then represent possible reconstructions of the evolution of those related sequences. If a sequence alignment occurs between two sequences, then it is called a pairwise alignment [3], [4], and the main goal is to find the similar or closely related parts between two sequences. If the alignment involves more than two sequences, then it is called a multiple sequence alignment and the main goal is to find the consensus parts among the sequences. For small lengths and small numbers of sequences, it is possible to create the alignment manually. However, efficient algorithms to align such sequences are essential for alignments with more than eight sequences [5].

MSA problems are solved using several different methods, such as classical, progressive, and iterative algorithms. These algorithms follow either global or local alignment strategies. In global alignments, sequences are aligned over their whole length. By contrast, local alignments identify regions of similarity within a sub sequence [6]. Local alignments are often preferable, but can be more difficult because of the additional challenge of identifying the regions of similarity. A general global alignment technique is the Needleman–Wunsch algorithm [3], which is based on dynamic programming. The Smith–Waterman algorithm [4] is a general local alignment method which is also based on dynamic programming. The dynamic programming (DP) approach [3] is good at finding the optimal alignment for two sequences. However, the complexity of this method grows significantly for three or more sequences [7]. Note that MSA is a combinatorial problem (NP-hard) where the computational effort becomes prohibitive with a large number of sequences [8]. The progressive alignment algorithm (tree-base algorithm), proposed by Feng and Doolittle [9], iteratively utilizes the method of Needleman and Wunsch [5] in order to obtain an MSA and to construct an evolutionary tree to depict the relationship between sequences. The progressive alignment algorithms align sequences according to the branching order of a guide tree. The difficulty with these methods is that they usually converge to local optima [5]. To overcome such a limitation, it is recommended to use an iterative or stochastic procedure [10]–[12].

Some of the proposed methods are based on local alignment. PIMA [13] uses local dynamic programming to align only the most conserved motifs. DIALIGN [14] uses a local alignment approach that constructs multiple sequence alignment based

Manuscript received November 4, 2010; revised February 20, 2011 and May 2, 2011; accepted July 9, 2011. Date of publication February 10, 2012; date of current version September 27, 2012.

The authors are with the School of Engineering and Information Technology, University of New South Wales at Australian Defense Force Academy, Canberra 2600, Australia (e-mail: fnrbitr@yahoo.com; r.sarker@adfa.edu.au; d.essam@adfa.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2011.2162849

段配段在保守基因很有效

on a segment to segment comparison, rather than the residue to residue comparison. This method is successful in highly conserved flanking core blocks, but is unreliable outside the conserved motifs [6]. T-Coffee [15] is a sensitive progressive alignment algorithm which combines information from global and local alignments in order to provide multiple sequence alignment. This method is fast but there is a possibility for it to become trapped at a local minima. There are also algorithms that are designed for sequence alignments of very large genomic regions up to mega bases long, such as AVID [16], BLASTZ [17], and MUMmer [18]. These methods effectively align closely related organisms, but have not been tested in alignments between more distant relatives [19]親緣近的物種有效

A good number of global alignment algorithms based on the progressive alignment method have been proposed to solve MSA problems such as MULTALIGN [20], MULTAL [21], PILEUP [22], and CLUSTALX [23]. MULTAL uses a sequential branching method to align the two closest sequences first, while then subsequently aligning the next closest sequences, and so on. MULTALIGN and PILEUP make the final alignments from the guide tree (the progressive alignment), which is constructed using unweighted pair group method using arithmetic averages (UPGMA) [24]. CLUSTAL W [5], based on a progressive approach, is a global method for multiple sequence alignments, which improves the local optimality issue of the progressive approach. The CLUSTAL W builds up the final alignments from a guide tree, which is calculated by a neighbor-joining (NJ) algorithm [25]. CLUSTAL W has one of the most sophisticated scoring systems, namely weighted sum of pair score, which considers sequence weighting and position dependent gap penalties. Although this approach is successful in a wide variety of cases, this method suffers from its greediness [15]. LAGAN and MLAGAN [19] are global alignment methods for large-scale pairwise and multiple genomic alignments. These methods are useful for both closely and distantly related organisms.

To overcome the limitations of the progressive alignment methods, researchers use either iterative or stochastic approaches. The iterative approach starts with an initial solution and then the current solution is improved using iterative steps. MUSCLE [26] solves MSA based on a progressive and iterative algorithm. It has three stages: draft progressive, improve progressive, and refinement. In each stage, a multiple sequence alignment is generated. Similarly MAFFT [27] is also based on a progressive and iterative algorithm. It uses a fast Fourier transform to identify homologous regions. ProbCons [28] is a probabilistic and consistency based algorithm. It computes posterior-probability matrices and expected accuracies for each pairwise comparison. Probcons achieves more accurate results than MUSCLE and MAFFT, but is slower than these algorithms [28]. PRRP [29] is another global alignment program which is based on a progressive and iterative approach. This approach is robust, but it is not guaranteed to find optimum solutions [15].

There are some iterative and stochastic approaches for MSA (for example, simulated annealing [30], [31] and evolutionary computation [32]–[36]). HMMT [37], based on a simulated annealing method, maximizes the probability for sequence alignment where the solution could become trapped in a local optima [38]. Evolutionary algorithms (EAs) are population based stochastic global search algorithms. When using EAs for MSA, an initial seed is generated by a progressive alignment method, and then the steps of an EA are applied to improve the similarities among the sequences. For example, MSA-EA [39] is used to improve the solution of the Clustal V [40] algorithm by initially generating one seed with Clustal V. This method works well for a large number of fully matched blocks, but performs poorly with only a few fully matched blocks [38]. There are some other genetic algorithm (GA) based methods for MSA, such as SAGA [36], GA-ACO [38], MSA-EC [41], MSA-GA [42], RBT-GA [43], and others.

In SAGA, the initial generation is generated randomly. In this algorithm, 22 different operators are used to gradually improve the fitness of the MSA. These operators are dynamically scheduled during the evolution process. The time complexity of SAGA is large, mainly due to the time required by the repeated use of the fitness function [41]. Shyu *et al.* [41] proposed two approaches for inferring MSA using GAs. In the first approach, GA was used to evolve an optimal guide tree. In this algorithm, the initial population of trees is generated randomly. In its crossover process, one portion of a binary tree (from the first parent) is connected to another portion of another binary tree (second parent) to generate a child binary tree. In its mutation process, some nodes are selected from a child tree and their upper edge is then connected to a randomly selected viable node. Shyu's second approach facilitates the optimization of a consensus sequence with a GA by using a vertically scalable encoding scheme, in which the number of iterations needed to find an optimal solution is approximately the same regardless of the number of sequences being aligned. Another algorithm, GA-ACO [38], combines ant colony optimization with GA to overcome the problem of becoming trapped in local optima. First, GA is run with a randomly generated initial population. Finally, ant colony optimization (ACO) was applied on the best alignment of the GA approach. MSA-GA is a simple GA based method with a different scoring function. To test this algorithm, the authors performed two sets of five runs for each of 28 test cases from the BAliBase 2.0 [44] dataset. In the first set, the initial population of this algorithm was generated only with pairwise alignments, but in the second set they consider alignments from both CLUSTAL W and the pairwise alignments as initial populations. RBT-GA is also a GA based method, combined with the rubber band technique (RBT), to find optimal protein sequence alignments [45], [46]. RBT is an iterative algorithm for sequence alignment using a DP table. The authors [43] solved 34 problems from reference sets 2 and 3 of the benchmark BAlibase 2.0 dataset. The experimental results showed that the overall performance of RBT-GA was better than the methods compared in that paper. In closing this discussion of GA for MSA, it should also be mentioned that to enhance the performance of GAs in solving MSA problems the local search methods are sometimes integrated with GAs [47]–[51]. GA + local search

There is no well-accepted theoretical model for sequence analysis [2]. An algorithm is accepted as a good method for

sequence alignment if it produces better fitness scores with respect to the benchmark datasets. From our literature review, it is observed that global alignment algorithms perform better than local alignment algorithms, and iterative and stochastic algorithms perform better than progressive approaches. Although the progressive alignment approaches are fast and deterministic, the main problem is that if any mistake is made early in the alignment process, then it cannot be corrected later. This is not an issue with iterative approaches. Also, most of these methods are stochastic, they are slower and the obtained results may vary between runs. However, the iterative methods are favorable for complex problems where either an alternative approach is unavailable or the quality of the alignment is more important than the computational cost. Therefore, the recent algorithms are based on global iterative and stochastic approaches. These methods are usually evaluated using a predefined objective function. Therefore, an objective function must be chosen in a way that is not only mathematically logical but that is also biologically meaningful. Designing an appropriate objective function is an ongoing research topic. Many iterative methods apply EAs. The EA based approaches have an important advantage over progressive methods in that the alignment component can be made independent of the objective function. This means that different fitness functions can be tested without making any adjustment to the alignment procedure, which makes them particularly attractive for testing new objective functions. Another useful advantage of these methods is that the computational duration can be shortened by parallelization. These points motivate us to apply EAs to solve MSA problems in this research.

In this paper, we have proposed a GA based approach, namely GAPAM, which starts with a DP distance table. In the DP distance table, the distance between two sequences is calculated from a pairwise alignment using DP. We have used this distance table to generate a guide tree. We have introduced two new techniques and have applied them on the guide tree to generate an initial population. To evaluate the MSA, we have employed the weighted sum of pair score as the fitness measure with the PAM250 [52] score matrix and the CLUSTAL W default gap penalties.

The performance of the proposed algorithm has been compared with some state-of-the-art GA and non-GA based methods in MSA-GA and RBT-GA papers, including SAGA, MSA-GA, RBT-GA, PRRP, CLUSTALX, CLUSTAL W, DIALIGN, HMMT, SB_PIMA, ML_PIMA, MULTALIGN, and PILEUP8. For comparison, we have considered only those benchmark datasets and algorithms that were considered in the papers reporting MSA-GA and RBT-GA. We have used the weighted sum of pair method (WSPM) for fitness evaluation. However, to allow us to compare with other methods, we have calculated the corresponding BAliscore of the best WSPM score. For comparison, the results of the 26 datasets solved by MSA-GA, and the alignment results of the 34 datasets solved by RBT-GA were taken from the published papers [42] and [43], respectively. However, the results of the other methods mentioned above were obtained from BAliBase 2.0 [44]. Based on the related BAliscore, GAPAM outperforms the GA and non-GA-based methods mentioned earlier.

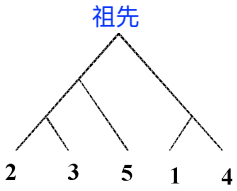

Fig. 1. Guide tree.

progressive錯誤無法被修正

This paper is organized as follows. After the introduction, a brief discussion about the tree based methods is given in Section II. Section III presents the steps of the proposed GAPAM method, Section IV presents a brief introduction of the test datasets, and the experimental study of the GAPAM and other methods are discussed in Section V. In Section VI, our conclusions are provided.

II. Tree-Based Method

A guide tree represents a hypothesis about the divergence of sequences from a common ancestor, and the assumption that this hypothesis guides the multiple alignment, which ideally should follow the same order as the order of sequence divergence as shown in Fig. 1.

A guide tree is calculated from a distance matrix [9]. The distance between two sequences can be measured using either an unaligned pair which is known as *k*-mer [53] or an aligned pair with DP [3] or with Kimura [54]. There are different methods to calculate the guide tree from the distance table, such as the NJ method [25], which produces unrooted trees with branch lengths proportional to the estimated divergence along each branch, and the UPGMA [24], which produces rooted trees with branch lengths proportional to the estimated divergence along each branch. The complexity of the neighbor-joining method is $O(N^4)$ which can be reduced to $O(N^3)$, whereas the complexity for UPGMA is $O(N^3)$ and can be reduced to $O(N^2)$ [26], where N is the number of sequences. Therefore, in this paper, we have used the UPGMA method to calculate guide trees. The complexity of the tree-base method is $O(N^3 + L^2)$, where L is the length of the sequences.

The following subsections discuss the distance calculation, the guide tree construction, and the output (multiple sequence alignment) of the guide tree.

A. Distance Calculation

1) *k-Mer Distance Calculation:* This distance is calculated from an unaligned pair of sequences. To describe this method [53], let us assume two sequences

$$X = \text{ABCDEF and } Y = \text{CDBCDA.}$$

長度為3的集合

We also consider that the length of a K-tuple, a segment length which is less than the sequences length, is 3. W_X and W_Y are the sets of all possible K-tuples of sequence X and sequence Y, respectively. Such as

$$W_X = \{\text{ABC, BCD, CDE, DEF, } \ldots\}$$

$$W_Y = \{\text{CDB, DBC, BCD, CDA, } \ldots\}.$$

iterator較慢但可產生較佳

對齊和適應度獨立

距離建構指引樹

So the resultant set from the above two sets is

$$W_{XY} = \{ABC, BCD, CDE, DEF, CDB, DBC, CDA, \ldots\}.$$

The length of W_{XY} is considered as m. C^X and C^Y are represented as the sets of common K-tuples between W_{XY} and W_X and also between W_{XY} and W_Y, respectively

$$C^X = \{1, 1, 1, 1, 0, 0, 0, \ldots\}$$

$$C^Y = \{0, 1, 0, 0, 1, 1, 1, \ldots\}.$$

The distance between two unaligned sequences is then calculated by the equation 相減為了求歧異，平方為了消正負

$$kmerdist = \sum_{i=1}^{m} \left(C_i^X - C_i^Y\right)^2. \quad (1)$$

The k-tuple (n-gram) based distance calculation is fast and easy to compute, hence this method requires little computation time [55]. However, the performance of this distance calculation depends on the size of the tuples. Shorter tuples contain less information and include more randomness, while longer tuples lengths contain more information and less randomness. Unfortunately though, as tuple length increases, vector size expands exponentially and thus becomes too large and computationally inefficient.

2) *Dynamic Distance Calculation:* In this case, the distance of each pair is calculated from a pairwise alignment. For a pairwise alignment problem, the well-known Needleman and Wunsch [3] or Smith–Waterman algorithm [4] can be employed. These methods use a dynamic programming table, which is filled beginning at the end of the sequences and attempts to match all possible pairs of residues according to a scoring scheme for matches, mismatches and gaps, thus generating a matrix of score values for all possible alignments between the two sequences. The matrix is built recursively according to (2), which ensures that the highest score identifies an optimal alignment. This matrix has dimensions (n, m), where n and m are the lengths of the two sequences, and the matrix is constructed from top to bottom; to reach a given position (i, j) in the matrix from a previous move, there are three possible paths. The first is a diagonal move with no gap penalty from position $(i-1, j-1)$, the next two a move from position $(i-1, j)$ to (i, j), and a move from position $(i, j-1)$ to (i, j), both have a gap penalty

$$F(i, j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \\ 0. \end{cases} \quad (2)$$

s為score, d為penalty

The value of $s(x_i, y_j)$ is determined from a substitution matrix (PAM [52] or BLOSUM [56]), in which a score is assigned to every possible substitution or conservation according to its probability in a biological system. In our research, we used the CLUSTAL W default gap penalty (i.e., d is -10 for the gap opening penalty).

The DP distance of each pair is calculated according to the mismatch of the pairwise alignment using (3). To construct the DP distance matrix (table), which shows the distance between all sequence pairs, the above step will continue for each pair of sequences

$$\text{Dynamic distance} = (mismatch)/(align\ length). \quad (3)$$

3) *Kimura Protein Distance Calculation:* Equation (4) is an alternate means to calculate the distance, and was developed based on the relationship between the observed amino acid substitutions and the actual (corrected) substitutions from PAM or BLOSUM. The PAM matrices are based on mutations observed throughout a global alignment, this includes both highly conserved and highly mutable regions. The BLOSUM matrices are based only on highly conserved regions in series of alignments forbidden to contain gaps. The match score is calculated by summing the number of exact matches. The partial matches between ambiguous symbols also contribute to the match score as fractional scores. The value of S is computed by dividing the match score by the number of positions scored. Gap positions are ignored, and only exact matches contribute to the match score [57]

$$S = (exact\_matches)/(positions\_scored)$$

$$D = 1 - S$$

$$\text{Distance} = -\ln(1 - D - 0.2\, D^2). \quad (4)$$

The most important consideration when calculating the distance between the aligned sequences, is that the sequences must be aligned properly [57]. This is because the aligned distance methods examine each pair of aligned sequences, symbol-by-symbol, and count the number of exact matches, partial matches and gap symbols. 正確對其很重要對於計算對齊分數

B. Guide Tree Construction and Multiple Alignment

The basic guide tree algorithm consists of the following stages.

Stage (i) Calculate the distances among all pairs.
Stage (ii) Prepare a distance matrix from Stage (i).
Stage(iii) Find a part of a guide tree by selecting the smallest distance pair from the distance matrix.

After these stages of the guide tree, the distance matrix is updated using (5), and then Stage (iii) is repeated to make a tree. This process continues until all sequences are combined and a complete guide tree is constructed as shown in Fig. 1

$$d_{ij} = \frac{1}{i_m} \sum_{m=1}^{i_m} \frac{\sum_{n=1}^{j_n} d_{irow(m)jcol(n)}}{j_n} \text{ (at } i \neq j), \quad d_{ij} = 0 \text{ (at } i = j). \quad (5)$$

距離表還會經過轉換成新距離表

Here, d_{ij} is the distance between the ith row and the jth column of a new distance matrix at $i > 1$ and $j > 1$. i_m is the number of elements in the ith row at $j = 1$, and the ith row elements are stored in the *irow* array. j_n is the number of elements in the jth column at $i = 1$ and also the jth column elements are stored in the *jcol* array. The value of $d_{irow(m)jcol(n)}$ is obtained from the original distance table (matrix).

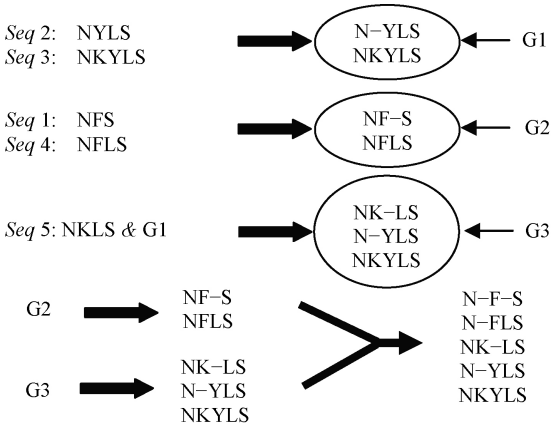

Fig. 2. Output (MSA) of guide tree.

Stage (iv) The sequences are progressively aligned according to the branching order in the guide tree as shown in Fig. 2. To present the output of the guide tree, let us assume five sequences as follows:

Seq 1 : NFS, *Seq* 2 : NYLS, *Seq* 3 : NKYLS, *Seq* 4 : NFLS, *Seq* 5 : NKLS.

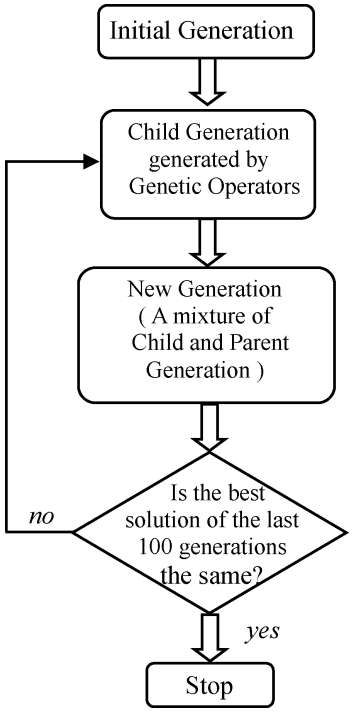

Fig. 3. Flowchart of GAPAM.

III. GAPAM: The Proposed Algorithm

The steps of our progressive alignment method using genetic algorithm (GAPAM) are: initial population, generation of child population by applying genetic operators, forming a new population for the next generation, and the stopping criteria. As we mentioned earlier, the solution of the progressive alignment method (tree-base) usually converges to a local optimum. Therefore, in the initial stage, we try to identify the local optima and their surrounding points. GA starts evolving from these individuals and leads the algorithm to find a better solution, which may be or may not be the global optimal solution. The flowchart of this method is shown in Fig. 3. The steps of this method are explained below.

A. *Initial Generation*

The aim of this step is to generate good initial solutions. The flowchart for generating the initial population is shown in Fig. 4 and the stages are described below.

Stage 1: GAPAM starts with a DP distance table. The DP distance is calculated from the pairwise alignment according to its mismatch. The guide tree is constructed from this table, which is referred to as TR1, and we generate the multiple sequence alignment (MSA1) from this guide tree.

Stage 2: In the second stage, the distance table is calculated from the multiple sequence alignment (MSA1), which is called the Kimura distance table. The Kimura distances are calculated from the aligned sequences. The second tree, TR2, is constructed from the Kimura distance table, and we then produce MSA2 as shown in Fig. 4.

Stage 3: In this stage, two mechanisms are implemented on the two trees, generated in Stage 1 and Stage 2, to generate 100 different trees. The first mechanism is to generate guide trees with randomly selected sequences and the second is shuffling the sequences inside that tree. The initial population produced by this method contains a set of multiple sequence alignments. Therefore, after receiving the set of guide trees, it needs to make a set of multiple sequence alignments. The functions of these mechanisms are explained below.

Mechanism 1: In this case, the sequence numbers are selected randomly from one tree (either TR1 or TR2). The selected sequences then make a new sub-tree with the same branching orders as the original one, and the non-selected sequences make a new sub-tree. Lastly, these two sub-trees are connected together to make a new tree. Fig. 5 shows the behavior of Mechanism 1.

Mechanism 2: In this case, two sequence numbers are selected randomly from one tree (either TR1 or TR2). Then, these two sequences exchange their positions to make a new tree. Fig. 6 shows the function of Mechanism 2.

B. *Fitness* 這邊fitness越大越好

The WSPM is commonly used as a fitness measure for multiple sequence alignments. Here, each column in an alignment is scored by summing the product of the scores of each pair of symbols and their pair weight. The score of the entire alignment is then summed over all column scores by using (6) and (7)

$$S = \sum_{l=1}^{L} S_l \text{ where } \quad S_l = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} W_{ij} \cos t(A_i, A_j). \qquad (6)$$

Here, S is the cost of the multiple alignments. L is the length (columns) of the alignment, S_l is the cost of the lth column of L length. N is the number of sequences, W_{ij} is the weight of

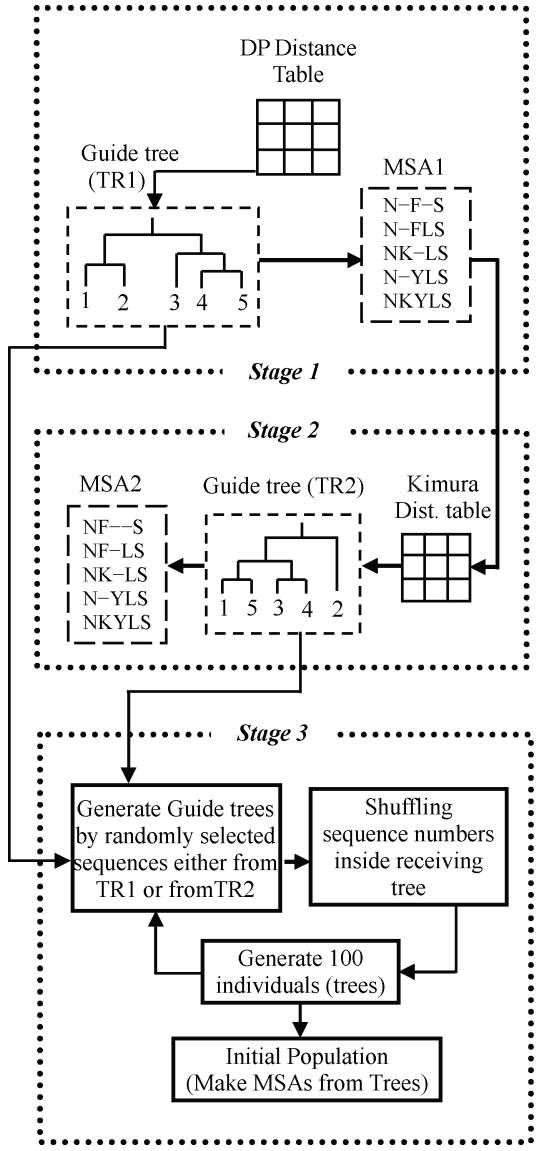

Fig. 4. Flowchart of initial generation.

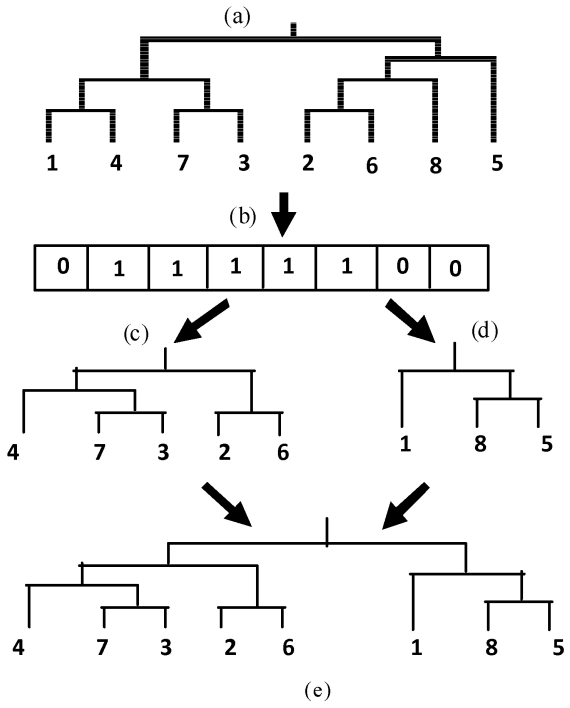

Fig. 5. (a) Guide tree. (b) **1** represents randomly selected sequence numbers from (a), and **0** represents unselected sequence numbers. (c) Subtree made by the selected sequences. (d) Subtree made by the remaining sequences of (a). (e) New guide tree made from (c) and (d).

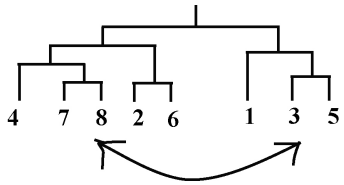

Fig. 6. Shuffling mechanism, sequence numbers 8 and 3 interchange their positions.

sequence i and j. In CLUSTAL W, the weight is calculated for each sequence and the pair weight is the product of the two sequence weight. $cost(A_i, A_j)$ is the alignment score between the two aligned sequences A_i and A_j. When $A_i \neq$ "-" and $A_j \neq$ "-" then $cost(A_i, A_j)$ is determined either from the percentage of acceptable point mutations (PAM) [52] or BLOSUM [56] matrix. Also when $A_i =$ "-" and $A_j =$ "-" then $cost(A_i, A_j) = 0$. Finally, the cost function $cost(A_i, A_j)$ includes the sum of the substitution costs of the insertion/deletions when $A_i \neq$ "-" and $A_j =$ "-" or $A_i =$ "-" and $A_j \neq$ "-," using a model with affine gap penalties as shown in score : match, blank, penalty

$$G = g + nx. \quad \text{開頭罰+延伸罰} \tag{7}$$

Here, G is the gap penalty, g is the cost of opening a gap, x is the cost of extending the gap by one, and n is the length of the gap.

In CLUSTAL W, the author used different weight matrices, which depend on the estimated divergence of the sequences to be aligned at each stage, and proposed dynamically changeable gap penalties to overcome the local minima issue. Therefore, in this research, the CLUSTAL W weighted scheme and the CLUSTAL W default gap penalties (gap opening penalty is -10 and the gap extension penalty is -0.20), and the PAM250 matrix, a mutation probability matrix, were considered with the WSPM fitness measure. Note that PAM250 is considered as a good general matrix for protein database searching. Also the PAM matrices have been developed based on global alignments. To calculate the weight of each sequence, we used the CLUSTAL W weight function.

C. Child Generation

For each individual in the initial population, the WSPM score is calculated, and the individuals are then sorted according to the descending order of their scores. To generate a child population of 100 individuals in any generation, the following three genetic operators are used:

1) single point crossover;
2) multiple point crossovers;
3) mutation.

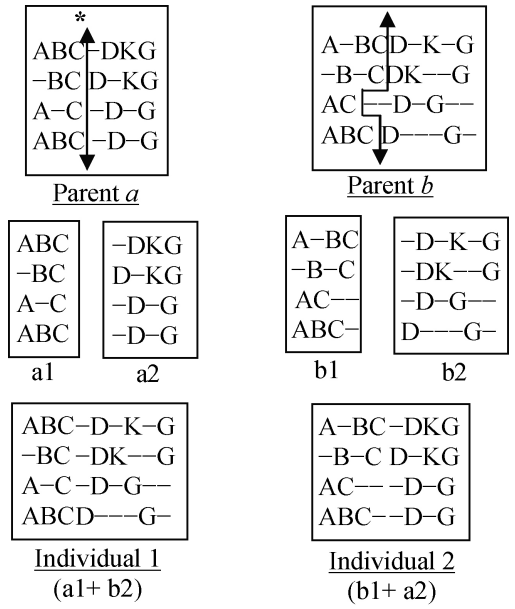

Fig. 7. Single point crossover.

1) *Single Point Crossover:* In this crossover, one individual is selected from the top 50% and another from the bottom 50% of the parent generation. The single point crossover [36] is implemented as shown in Fig. 7. Its procedure is that first a column position is selected randomly as shown with a "*" in Fig. 7. The parent having the better score is then divided vertically at that column. Let us assume that parent *a* has the better score column, so that this parent is separated vertically into two pieces. The second parent *b* is also divided into two pieces in such a way that each row of the first piece (and hence also the second piece) has the same number of elements as the first piece (and hence also the second piece) of the first parent. These pieces of these two parents are then exchanged and merged together to generate two new individuals as shown in Fig. 7. However, only the better new individual is considered as a child.

2) *Multiple Point Crossovers:* In this crossover, two parents are selected, one from the top 50% and another from the bottom 50% of the parent generation. For multiple crossovers, each parent is divided into three pieces. The different pieces from these parents are then exchanged and merged together to generate two new individuals. However, the better one will be taken as a child. The crossover is implemented in two steps as described below.

Step 1: To cut the first piece effectively, we compare the scores of the first 25% of columns for both parents. The parent having the better score is divided vertically at that column. The other parent is divided using the mechanism introduced in the single point crossover, as can be seen in Fig. 8.

前25%好者

Step 2: We now have two pieces of each parent from Step 1. To create another piece, we follow the same procedure of Step 1, with considering the last 25% of columns (see Fig. 8). This gives us three pieces for each parent. To complete the crossover, the middle pieces are exchanged between the parents, and then all three pieces are merged together to generate two new individuals as shown in Fig. 8.

後25%好者

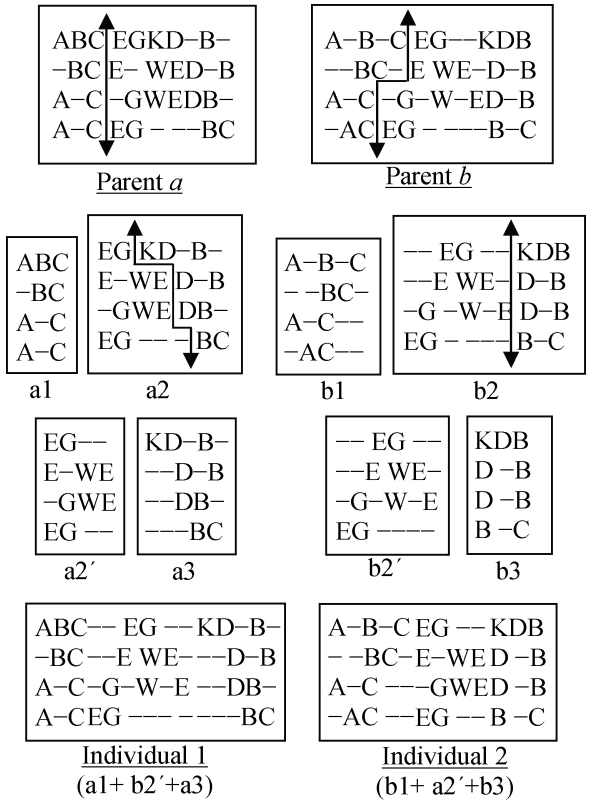

Fig. 8. Multiple point crossovers.

In Fig. 8, the lengths (columns) of the two parents (parent *a* and parent *b*) are 10 and 12, respectively. The first 25% of the columns of the first parent are two and a half (2.5) columns. In that case, we considered three columns. In the first step, parent *a* has the better score in the first 25% columns. Therefore, Parent *a* is divided first and Parent *b* is tailored according to Parent *a*. After the division, we have two pieces from each parent (a1 and a2 from Parent *a*, b1 and b2 from Parent *b*). In the second step, Parent *b* has the better score in the last 25% of columns. Therefore, Parent *b* is divided first and Parent *a* is tailored accordingly. This division provides two new pieces for each parent (a2′ and a3 from a2, b2′ and b3 from b2). Next, two new individuals are generated by connecting the pieces as (a1 + b2′ + a3) and (b1 + a2′ + b3). From these two individuals, the better one is selected as a child.

10*25%=2.5

3) *Mutation:* One individual (MSA) is randomly selected from the whole populations. From this MSA, the distance among sequences is calculated and stored in a distance table. The new guide tree is constructed from this calculated distance. In the new guide tree, the sequence numbers are shuffled to find a better guide tree and the MSA of the new guide tree is considered as a mutated child (see Fig. 9).

D. New Generation

To form the new generation, we have used the $\mu+\gamma$ selection strategy, where the multiple sequence alignments from the parent (μ) and child (γ) generations compete based on their objective fitness scores. In this research, the best 50% of the

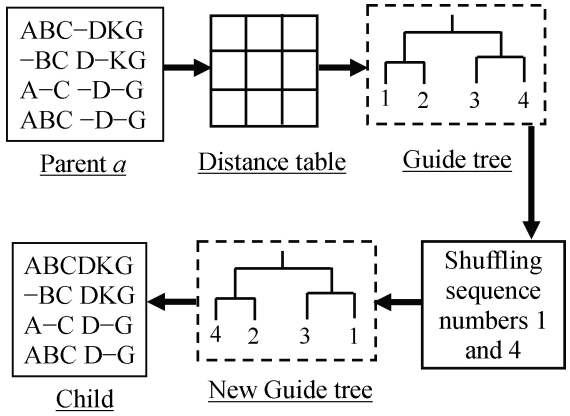

Fig. 9. Mutation.

parents and the best 50% of the children are merged together while ensuring that there is no duplication of individuals. We have also experimented with other splits, such as 40–60 (parent–child) and 60–40. The results of those experiments showed that the 50–50 split outperforms the 40–60 and 60–40 splits with an average improvement of 4.38% and 7.66%, respectively. Therefore, we have chosen this mix based on our experimental observations, which ensures a better balance between exploration and exploitation. The new generation is then considered as the parent population that is used to continue the evolution process of GAPAM.

E. Termination Condition

如果停滞100代，就結束

The best solution in each generation is recorded. If the best solution remains the same in 100 consecutive generations, the algorithm will be terminated. This termination condition was based on our experimental observations. We tested our algorithm for up to 300 generations after getting the best solution, and we observed that the best solution was hardly changed, and that the variation of the average solution per generation was also small. The computational complexity of the GAPAM method was $(N^3 + L^2)$, where L is the length of the sequence and N is the number of sequences.

IV. Test Data Sets

In order to evaluate our proposed approach, we have solved a good number of test datasets from the benchmark BAliBase alignment database. The original BAliBase version 1.0 [58] consists of 142 reference alignments with over 1000 sequences. BAliBase version 2.0 [44] is an improved version, extended from version 1 with 167 reference alignments, to over 2100 sequences. BAliBase version 2.0 contains eight reference sets. Each reference has a variety of alignment problems. Reference 1 contains small numbers of equidistant sequences. The orphan or unrelated sequences are considered in reference 2. Reference 3 contains a pair of divergent subfamilies where the two groups are less than 25% identical. Reference 4 contains long terminal extensions, and reference 5 contains large internal insertions and deletions. Lastly, reference sets 6–8 contain test case problems where the sequences are repeated and the domains are inverted.

V. Experimental Study

In this section, we have first analyzed the performance of the proposed GAPAM algorithm, and then we have compared our algorithm with other well-known methods. In this research, we have analyzed our results based on ten independent runs. In comparison, MSA-GA and RBT-GA used five and ten independent runs, respectively.

A. Experimental Analysis

In this section, we have reported a parametric analysis, analyzed the effect of both search operators and the initial population on the performance of the proposed algorithm, and discussed the computational effort required.

1) *Selection of Parameters:* In the proposed GAPAM algorithm, we have used two basic search operators: crossover and mutation. In order to determine the best mix of the probabilities of crossover and mutation, we have carried out five different experiments, using ten randomly selected BaliBase datasets (version 2.0) [44], as follows:

a) 100% crossover (100-crc);
b) 60% crossover and 40% mutation (60–40 crc-mu);
c) 50% crossover and 50% mutation (50–50 crc-mu);
d) 40% crossover and 60% mutation (40–60 crc-mu);
e) 100% mutation (100-mu).

For each of the ten datasets, the algorithm was executed for ten independent runs. For each dataset, the best WSPM score out of the ten runs was recorded and the corresponding BAliscore was reported in Table I, where the bold face value represents the best score. From this table, it is observed that GAPAM with the 50% crossover and 50% mutation option has obtained the best solutions for seven out of ten datasets, the 60% crossover and 40% mutation for two and 40% crossover and 60% mutation for one but the solution is the same as that of 50% crossover and 50% mutation. The options 100% crossover has achieved the best solution in one test case, however the option 100% mutation has not achieved any good quality solutions. The solutions obtained by the 50% crossover and 50% mutation for the other three datasets are close to the best scores. Therefore, we can conclude that GAPAM has achieved overall better performance for these test datasets when the rate of crossover and mutation were selected as 50% for each. When comparing the computational times, 100% crossover is the least expensive and 100% mutation is the most expensive option. The option 50% crossover and 50% mutation is in the middle place. The options 60% crossover and 40% mutation and 40% crossover and 60% mutation are in second and fourth place, respectively. However, it is worth recalling that although the 100% crossover is the least expensive option, it was only the best in one test case. Considering both the quality of solutions and the computational time required, we can say that 50% crossover and 50-mutation is the best option.

2) *Effect of Operators and Initial Population:* The proposed algorithm GAPAM uses an improved initial population and new genetic operators that contribute to it performing better than other algorithms. To analyze the effect of these two components on the algorithm's performance, we have

TABLE I
PARAMETER ANALYSIS

Name of Datasets		100% Crossover		60% Crossover and 40% Mutation		50% Crossover and 50% Mutation		40% Crossover and 60% Mutation		100% Mutation	
		Corresponding BAliscore of Best WSPM Score	Average Computation Times (s)	Corresponding BAliscore of Best WSPM Score	Average Computation Times (s)	Corresponding BAliscore of Best WSPM Score	Average Computation Times (s)	Corresponding BAliscore of Best WSPM Score	Average Computation Times (s)	Corresponding BAliscore of Best WSPM Score	Average Computation Times (s)
Ref. 2	1havA	0.842	164.20	0.869	796.40	**0.879**	610.36	0.875	931.40	0.847	1110.42
	1uky	0.726	306.00	0.817	681.09	**0.808**	837.46	0.817	1207.45	0.746	2334.56
	2hsdA	0.765	271.20	**0.816**	665.40	0.796	869.25	0.785	1236.20	0.790	1783.41
	2pia	**0.829**	205.20	0.799	1090.40	0.826	1337.33	0.797	1976.56	0.795	2144.20
	1pamA	0.825	469.20	0.814	1662.25	**0.860**	1865.51	0.849	2570.31	0.851	3191.15
Ref. 3	1wit	0.775	96.20	**0.788**	175.36	0.758	253.12	0.761	669.20	0.736	1035.30
	1uky	0.464	314.40	0.440	729.31	**0.468**	818.44	0.447	1226.20	0.410	2270.40
	kinase	0.808	252.00	0.811	713.20	**0.828**	944.35	0.823	1437.20	0.803	2343.20
	1pamA	0.795	516.00	0.834	1488.20	**0.835**	1792.10	**0.835**	2486.20	0.775	2832.21
	4enl	0.679	394.12	0.837	1465.40	**0.800**	2011.74	0.790	1518.40	0.736	2011.21

TABLE II
PERFORMANCE TEST OF THE GENETIC OPERATORS AND THE INITIAL GENERATION OF GAPAM METHOD BY COMPARING WITH THE SOLUTION OF HILL CLIMBING AND GAPAM WITH THE RANDOMLY GENERATED INITIAL POPULATION RESPECTIVELY

Name of Datasets		GAPAM (BAliscore)	Hill Climbing (BAliscore)	GAPAM with Randomly Generated Initial Population (BAliscore)
Ref. 2	1havA	**0.879**	0.810	0.813
	1uky	**0.808**	0.723	0.750
	2hsdA	**0.796**	0.718	0.764
	2pia	**0.826**	0.742	0.703
	1pamA	**0.860**	0.746	0.826
Ref. 3	1wit	**0.758**	0.611	0.676
	1uky	**0.468**	0.403	0.458
	kinase	**0.828**	0.787	0.771
	1pamA	**0.835**	0.723	0.770
	4enl	**0.800**	0.704	0.716
Average score		**0.786**	0.697	0.725
Average (%) improvement of GAPAM over			12.79	8.43

designed two sets of experiments. In the first set, GAPAM was run with a *randomly generated* initial population (instead of our improved initial population), and the second set used a hill climbing approach (for searching instead of GAPAM) starting from the improved initial population. WSPM was used as the fitness measure. We have used ten BAliBase datasets for these experiments (five from reference set 2 and five from the reference set 3). Each dataset was run with GAPAM (with the two different configurations) for ten independent runs. Based on the corresponding BAliscore of the best WSPM solution found, GAPAM with improved initial population outperforms GAPAM with initial random population for all datasets and shows an average improvement of 8.43%. The results are similar with the hill climbing approach, where GAPAM shows an improvement of 12.79%. The first set of experiments thus proves the superiority of our proposed initial population, and the second set clearly demonstrates the ability of our proposed genetic search operators to outperform simple hill-climbing. The details of these experiments are reported in Table II.

3) *Computational Effort and Convergence:* The computational time required for finding good multiple sequence alignments is dependent on the sequence length, the number of sequences, and the similarities of the sequences. In addition, the choice of algorithmic parameters also plays an important role. We have tried to develop a relationship between the computational time required (with our algorithm) and the sequence length and sequence numbers. Although it is hard to make any firm conclusion based on linear/nonlinear regression analysis, it showed an approximately polynomial relationship with degree of two. We must mention here that we have tried linear and other combinations of higher order polynomial functions.

To show the convergence behavior of our algorithm, we have plotted the best and the average WSPM scores against the number of generations. As for example, three such plots (for one specific run) for three datasets from reference set 3 are presented in Fig. 10. These graphs showed that our algorithm improved both the best and the average scores very rapidly at the initial stage of the search process and that the best score then converged to a solution. This is the type of pattern we expect from good search algorithms. As of the plots, although the average scores do not converge, the rate of improvement for the best score in the later generations of the algorithm is insignificant.

B. Solution Quality Assessment

To judge the quality of the solutions produced by our algorithm, we have considered only those benchmark datasets and algorithms that were considered in the papers reporting MSA-GA and RBT-GA, and that used BAliscore (an open source program of the BAliBase benchmark) to measure the accuracy of the solutions. The authors of MSA-GA considered the best solution of five runs for each dataset and reported the BAliscore. Moreover, the authors of RBT-GA also reported the best solution with BAliscore of ten runs. In our algorithm, we considered ten independent runs of each dataset and have used the corresponding BAliscore of the best found WSPM solution. BAliscore scores a solution (multiple sequence alignment) between 0.0 and 1.0. If the solution is identical with the corresponding manually created reference alignment then the score is 1.0. If nothing matches with the reference alignment then the score is 0.0. However, if some parts match with the reference alignment then the score is lower than 1.0 but greater than 0.0.

In MSA-GA, the authors considered 28 test datasets from reference sets 1 to 5 and 8. Among them, 18 datasets were from reference 1 and two were from each of the other reference datasets. However, currently BAliscore does not work for

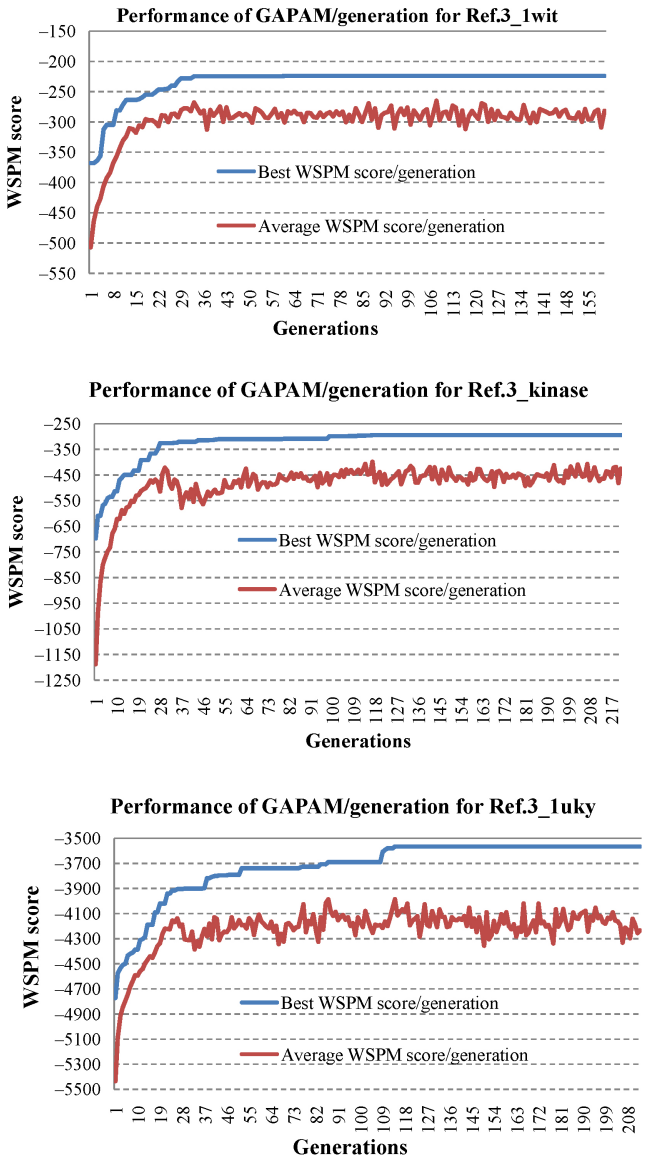

Fig. 10. Graphical presentations of the performance of the GAPAM method w.r.t. the best and average WSPM score per generation.

reference set 8. This is because of insufficient information supplied either by the reference alignment file or by the annotation file. Therefore, we excluded the two datasets of reference 8, thus leaving 26 for comparison. In RBT-GA, the author considered all 23 test datasets of reference 2, and 11 out of 12 from reference 3. In total, we considered 56 test datasets including 18 from [1], 23 (all) from reference 2, 11 from reference 3, and 2 from each of references 4 and 5. All these datasets belong to the BAliBase 2.0 benchmark datasets.

1) *Problem Solving with GAPAM:* For each of the 56 datasets, we have executed our algorithm for ten independent runs and recorded the best, worst, and average WSPM scores with standard deviation, and the corresponding BAliscore of the best WSPM score in Table III. The WSPM scores could be either positive or negative, as it depends on the level of similarity among the residues in the sequences. This is because, if the residues among the comparable sequences are similar, or partially similar, it needs a small number of null ("-") symbols to make an alignment of the sequences. In this case, the WSPM score of this alignment is positive. On the other hand, if the dissimilar parts among the sequences are high, a large number of null symbols are added to the alignment. In this case, the WSPM score becomes negative because of gap penalties. Note that high positive values and low negative values are considered as good scores. We must also mention here that the average scores and hence the standard deviations in the ten runs were not very different.

For comparisons with other methods, we have taken the results of those methods from their published literature [6], [42], [43], as discussed below. However, it is not possible for us to compare our results with MUSCLE [26] and MAFFT [27], because the datasets and/or the fitness measure used by these two algorithms do not match with the same of our algorithm. For example, these algorithms do not provide BAliscore results for the individual problems of the BAliBase 2.0 test sets. Instead, the authors of MUSCLE reported the average Q score of the BAliBase 2.0 datasets, while the authors of MAFFT considered the datasets from BAliBase 3.0.

2) *Comparing GAPAM with MSA-GA and Other Methods:* The authors of MSA-GA [42] selected 28 test cases from reference sets 1 to 5 and 8. As discussed earlier, we have considered 26 out of these 28 test cases. The results are provided in Table IV and plotted in Figs. 11 and 12.

In Table IV, the bold face data represents the best performing scores among the methods. From Table IV and Fig. 11, it is observed that GAPAM obtained more accurate solutions in 19 test cases out of 26, whereas MSA-GA was more accurate in two, MSA-GA w/prealign in one, SAGA in four, and CLUSTAL W in one. In seven test cases, where GAPAM did not achieve the best solutions, the solutions are close to the best solutions of the other methods reported in this table.

To evaluate the overall performance of all the methods reported in Table IV, the average scores of 26 test cases were calculated and reported in the bottom row. The average score of GAPAM for 26 datasets is the best of all the algorithms. From the experimental results, we can claim that GAPAM has better performance on these 26 test cases.

3) *Comparing GAPAM with RBT-GA:* We have considered all of the 34 datasets solved by RBT-GA [43]. We have taken the approximate results of RBT-GA as reported in the paper [43]. The summary of the experimental results of reference sets 2 and 3 is presented in Tables V and VI and is plotted in Figs. 13 and 14, and Figs. 15 and 16, respectively.

a) *Performance of GAPAM in Reference 2:* The 23 datasets in this reference are significantly different in lengths and numbers of their sequence. They also contain what is called "orphan sequences." GAPAM performed differently with different datasets. To judge the performance of GAPAM with respect to BAliscore, we have compared with SAGA, RBT-GA, PRRP, CLUSTALX, DIALIGN, HMMT, SB_PIMA, ML_PIMA, MULTALIGN, and PILEUP8. Table V and Fig. 13 show that for the 23 test cases, GAPAM has successfully found more accurate solutions than the others in 15 test cases, RBT-GA in 5, PRRP in two, and CLUSTALX in one. In eight test cases, where GAPAM could not obtain the best solutions, they were close to the best solutions.

TABLE III
SUMMARY OF THE TEST RESULTS OF GAPAM METHOD

Name of Datasets		Sequence Number	Sequence Length	With WSPM				
				Best Score	Worst Score	Avg. Score	Std	Corresponding BAliscore
Ref. 1	1idy	5	58	65.93	42.45	49.97	7.13	0.565
	1tvxA	4	69	18.387	13.56	16.69	5.29	0.316
	1uky	4	220	−5.52	−7.25	−6.35	2.86	0.402
	Kinase	5	276	−78.27	−115.28	−89.83	10.67	0.487
	1ped	3	374	23.44	5.94	12.51	6.48	0.498
	2myr	4	474	−83.71	−89.32	−84.34	1.87	0.317
	1ycc	4	116	−27.419	−2.581	−10.42	8.96	0.845
	3cyr	4	109	32.37	23.75	26.78	2.56	0.911
	1ad2	4	213	41.26	37.96	39.43	1.11	0.956
	1ldg	4	675	46.72	34.83	42.07	4.15	0.963
	1fieA	5	442	274.45	164.34	213.81	39.57	0.963
	1sesA	5	63	235.37	140.94	183.36	33.94	0.982
	1krn	4	82	70.62	62.29	65.59	5.20	0.960
	2fxb	5	63	150.14	89.70	116.82	21.69	0.970
	1amk	5	258	73.50	44.01	57.25	10.6	0.998
	1ar5A	4	203	40.99	24.53	31.92	5.92	0.974
	1gpb	5	828	836.78	742.06	794.76	33.56	0.983
	1taq	5	928	614.13	535.85	579.98	23.91	0.945
Ref. 2	1aboA	15	80	−377.456	−449.87	−454.63	37.86	0.796
	1idy	19	60	408.98	347.68	380.81	19.75	0.989
	1csy	19	99	−69.93	−151.77	−101.09	23.45	0.764
	1r69	20	76	−28.937	−74.86	−50.94	17.23	0.965
	1tvxA	16	69	334.38	300.64	315.46	10.04	0.92
	1tgxA	19	71	342.38	262.79	307.97	25.94	0.878
	1ubi	19	60	36.30	−40.42	4.93	41.54	0.767
	1wit	20	106	−120.21	−256.97	−183.60	49.45	0.851
	2trx	19	94	903.23	814.55	855.29	30.21	0.986
	1sbp	16	262	−19.79	−87.38	−75.19	24.93	0.765
	1havA	16	242	49.62	20.357	33.57	11.95	0.879
	1uky	23	225	−84.92	−180.95	−121.98	35.64	0.808
	2hsdA	20	255	−389.79	−462.57	−443.99	30.55	0.796
	2pia	16	294	−146.38	−282.48	−223.91	30.26	0.828
	3grs	15	237	−142.16	−288.47	−210.66	36.37	0.746
	Kinase	18	287	−191.16	−255.51	−224.46	2072	0.799
	1ajsA	18	389	1956.94	1873.33	1920.89	30.95	0.899
	1cpt	15	434	−435.69	−533.92	−490.22	37.40	0.875
	1lvl	23	473	−826.15	−1030.76	−916.93	88.66	0.781
	1pamA	18	511	−974.64	−1040.92	−1019.11	26.78	0.860
	1ped	18	388	1940.82	1811.97	1862.50	34.46	0.912
	2myr	17	482	13970.32	13788.35	13865.22	64.49	0.822
	4enl	17	440	1386.81	1222.06	1299.04	54.08	0.896
Ref. 3	1idy	27	60	−512.34	−677.41	−588.37	70.85	0.601
	1r69	23	78	−1103.34	−1240.69	−1174.5	50.35	0.709
	1ubi	22	97	−959.62	−1026.13	−1004.26	25.59	0.386
	1wit	19	102	−223.56	−357.01	−263.99	34.67	0.758
	1uky	24	220	−3565.98	−3710.55	−3662.99	46.78	0.468
	kinase	18	287	−294.62	−412.63	−357.70	36.54	0.828
	1ajsA	28	396	−4199.49	−4295.81	−4250.98	38.87	0.311
	1pamA	19	511	−2106.98	−2218.21	−2163.54	49.25	0.835
	1ped	21	388	−1199.41	−1304.44	−1251.38	38.58	0.813
	2myr	21	482	−6498.07	−6645.10	−6574.31	54.84	0.513
	4enl	19	427	−45.86	−120.70	−72.34	23.33	0.800
Ref. 4	1dynA	6	848	−101338.46	−101396.19	−101377.23	24.88	0.033
	Kinase2	7	468	−25744.36	−25839.97	−25792.01	26.42	0.384
Ref. 5	2cba	8	328	−987.14	−1095.04	−1047.40	44.01	0.852
	S51	15	301	−2553.28	−2706.59	−2648.79	52.83	0.835

TABLE IV
EXPERIMENTS ON SELECTED DATASETS OF MSA-GA

Name of Datasets		GAPAM	MSA-GA	MSA-GA w/prealign	SAGA	CLUSTAL W
Ref. 1	1idy	**0.565**	0.427	0.438	0.342	0.500
	1tvxA	**0.316**	0.295	0.209	0.278	0.042
	1uky	0.402	0.443	0.405	**0.672**	0.392
	Kinase	0.487	0.295	0.488	**0.862**	0.479
	1ped	0.498	0.501	0.687	**0.746**	0.592
	2myr	**0.317**	0.212	0.302	0.285	0.296
	1ycc	**0.845**	0.650	0.653	0.837	0.643
	3cyr	**0.911**	0.772	0.789	0.908	0.767
	1ad2	**0.956**	0.821	0.845	0.917	0.773
	1ldg	0.963	0.895	0.922	**0.989**	0.880
	1fieA	**0.963**	0.843	0.942	0.947	0.932
	1sesA	**0.982**	0.620	0.913	0.954	0.913
	1krn	**0.960**	0.908	0.895	0.993	0.895
	2fxb	0.970	0.941	**0.985**	0.951	**0.985**
	1amk	**0.998**	0.965	0.959	0.997	0.945
	1ar5A	**0.974**	0.812	0.946	0.971	0.946
	1gpb	**0.983**	0.868	0.948	0.982	0.947
	1taq	**0.945**	0.525	0.826	0.931	0.826
Ref. 2	2pia	**0.877**	0.761	0.768	0.763	0.766
	1pamA	**0.859**	0.755	0.758	0.623	0.757
Ref. 3	Kinase	**0.825**	0.58	0.619	0.758	0.619
	1pamA	**0.835**	0.703	0.744	0.579	0.743
Ref. 4	1dynA	0.033	**0.038**	0.034	0.000	0.000
	Kinase2	0.384	**0.71**	0.635	0.364	0.630
Ref. 5	2cba	**0.852**	0.422	0.621	0.767	0.628
	S51	**0.835**	0.528	0.73	0.831	0.75
Average Score		**0.751**	0.528	0.695	0.740	0.679

TABLE V
EXPERIMENTS ON REFERENCE 2 DATASETS OF BALIBASE 2.0

Name of Datasets		PRRP	CLUSTAL X	SAGA	DIALI	HMMT	SB_PIMA	ML_PIMA	MULTALIGN GN	PILEUP8	RBTGA	GAPAM
Ref. 2	1aboA	0.256	0.65	0.489	0.384	0.724	0.391	0.22	0.528	0.000	**0.812**	0.796
	1idy	0.37	0.515	0.548	0.000	0.353	0.000	0.000	0.401	0.000	**0.997**	0.989
	1csy	0.35	0.154	0.154	0.000	0.000	0.000	0.000	0.154	0.114	0.735	**0.764**
	1r69	0.675	0.675	0.475	0.675	0.000	0.675	0.675	0.675	0.45	0.9	**0.965**
	1tvxA	0.207	0.552	0.448	0.000	0.276	0.241	0.241	0.138	0.345	0.891	**0.92**
	1tgxA	0.695	0.727	0.773	0.63	0.622	0.678	0.543	0.696	0.318	0.835	**0.878**
	1ubi	0.056	0.482	0.492	0.000	0.053	0.129	0.129	0.000	0.000	**0.795**	0.767
	1wit	0.76	0.557	0.694	0.724	0.641	0.469	0.463	0.5	0.476	0.825	**0.851**
	2trx	0.87	0.87	0.87	0.734	0.739	0.85	0.702	0.87	0.87	0.982	**0.986**
	1sbp	0.231	0.217	0.374	0.043	0.214	0.043	0.054	0.186	0.177	**0.778**	0.765
	1havA	0.52	0.48	0.448	0.000	0.194	0.259	0.238	0.5	0.493	0.792	**0.879**
	1uky	0.351	0.656	0.476	0.216	0.395	0.256	0.306	0.585	0.562	0.625	**0.808**
	2hsdA	0.404	0.484	0.498	0.262	0.423	0.39	0.561	0.593	0.278	0.745	**0.796**
	2pia	0.767	0.752	0.763	0.612	0.647	0.73	0.695	0.765	0.766	0.730	**0.826**
	3grs	0.363	0.192	0.282	0.350	0.141	0.183	0.211	0.192	0.159	**0.755**	0.746
	Kinase	0.896	0.848	0.867	0.692	0.749	0.755	0.651	0.83	0.799	0.712	0.799
	1ajsA	0.227	0.324	0.311	0.000	0.242	0.000	0.000	0.311	0.227	0.892	**0.899**
	1cpt	0.821	0.66	0.776	0.425	0.388	0.184	0.277	0.777	0.688	0.584	0.875
	1lvl	0.772	0.746	0.726	0.783	0.539	0.62	0.688	0.614	0.678	0.567	0.781
	1pamA	0.711	0.761	0.623	0.576	0.53	0.393	0.386	0.566	0.702	0.66	**0.86**
	1ped	0.881	0.834	0.835	0.773	0.696	0.651	0.647	0.741	0.749	0.78	**0.912**
	2myr	0.582	0.904	0.825	0.84	0.443	0.727	0.75	0.894	0.786	0.675	0.822
	4enl	0.668	0.375	0.739	0.122	0.213	0.096	0.092	0.384	0.224	0.812	0.896
Average Score		0.541	0.583	0.586	0.384	0.401	0.379	0.371	0.517	0.429	0.777	**0.851**

TABLE VI
EXPERIMENTS ON REFERENCE 3 DATASETS OF BALIBASE 2.0

Name of Datasets		PRRP	CLUSTAL X	SAGA	DIALI	HMMT	SB_PIMA	ML_PIMA	MULTALIGN GN	PILEUP8	RBTGA	GAPAM
Ref. 3	1idy	0.000	0.273	0.364	0.000	0.227	0.000	0.000	0.045	0.000	0.546	**0.601**
	1r69	**0.905**	0.524	0.524	0.524	0.000	0.000	**0.905**	0.000	0.000	0.374	0.709
	1ubi	0.415	0.146	**0.585**	0.000	0.366	0.000	0.000	0.000	0.268	0.31	0.386
	1wit	0.742	0.565	0.484	0.500	0.323	0.645	0.323	0.242	0.210	**0.78**	0.758
	1uky	0.139	0.130	0.269	0.139	0.037	0.083	0.148	0.241	0.083	0.35	**0.468**
	kinase	0.783	0.720	0.758	0.650	0.478	0.541	0.682	0.688	0.599	0.697	**0.828**
	1ajsA	0.128	0.163	0.186	0.000	0.006	0.000	0.000	0.000	0.110	0.18	**0.311**
	1pamA	0.683	0.678	0.579	0.683	0.169	0.546	0.590	0.546	0.754	0.525	**0.835**
	1ped	0.679	0.627	0.646	0.641	0.172	0.450	0.507	0.665	0.722	0.425	**0.775**
	2myr	0.646	0.538	0.494	0.272	0.101	0.278	0.494	0.253	0.310	0.33	**0.813**
	4enl	0.736	0.547	0.672	0.050	0.050	0.393	0.438	0.652	0.498	0.68	**0.8**
Average Score		0.532	0.446	0.506	0.314	0.175	0.267	0.372	0.303	0.323	0.472	**0.662**

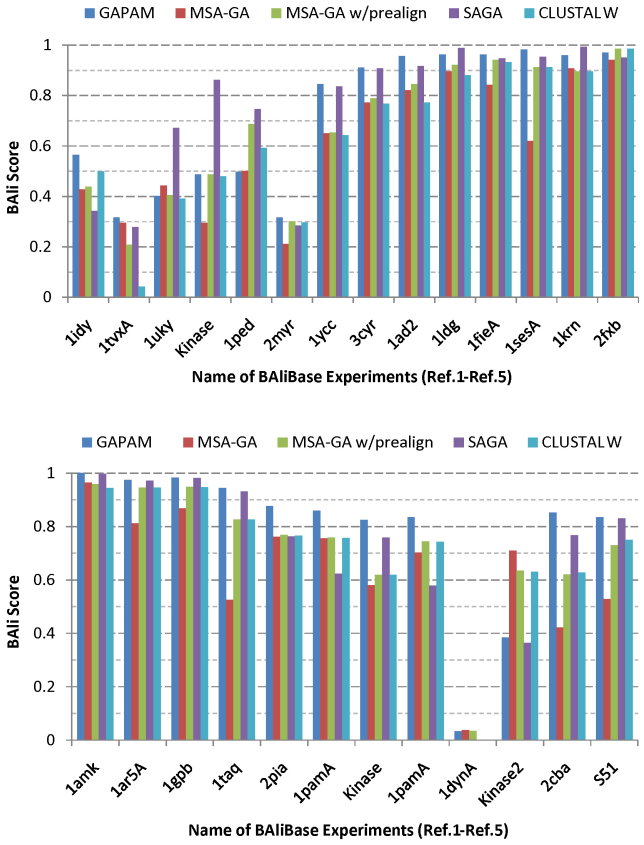

Fig. 11. Graphical presentations of the experimental results on MSA-GA selected datasets.

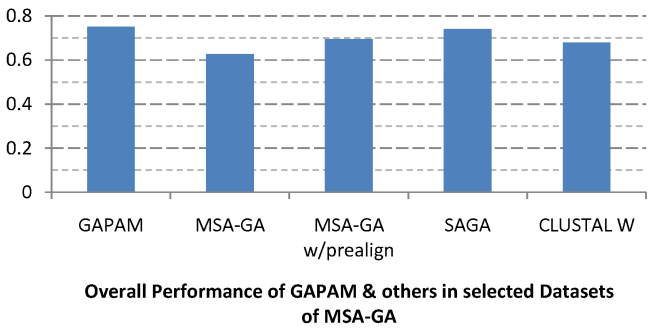

Fig. 12. Overall performance of all methods in MSA-GA selected datasets.

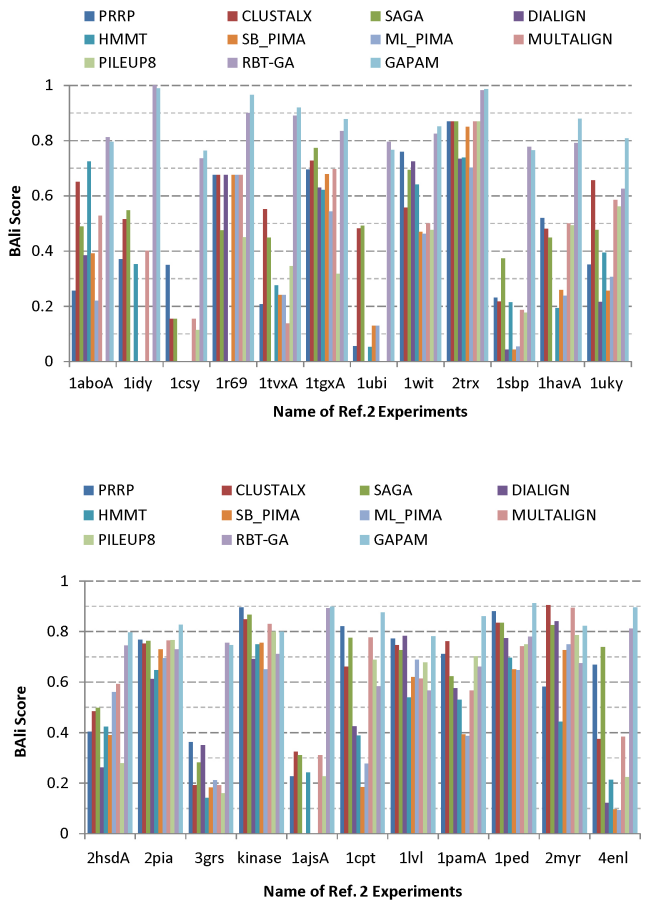

Fig. 13. Graphical presentations of the experimental results on reference 2 datasets.

The overall performance, from the average score in Table V, of all methods for reference 2 is shown in Fig. 14. This figure shows that GAPAM achieved higher average accuracy than all of the other methods considered in this section. GAPAM performed better for almost all test cases in reference 2.

b) *Performance of GAPAM in Reference 3:* Reference 3 contains sub-groups of sequences where the residue identities between groups are less than 25%. In this paper, we considered 11 test cases out of 12, and the experimental results that are illustrated in Table VI and Fig. 15 show that GAPAM found

TABLE VII
WILCOXON SIGNED RANK TEST RESULTS FOR THE GAPAM AND OTHER METHODS

Algorithms	Comparing the GAPAM (With Respect to the BAliscore)				
	W+	W−	P	GAPAM is Significant If P< 0.05	Hypothesis Test Decision (Null Hypothesis)
MSA-GA	22	4	0.0004239	Yes	Reject
MSA-GA w/prealign	20	6	0.003088	Yes	Reject
SAGA (with MSA-GA test sets)	21	5	0.055	No	**Retain**
CLUSTAL W	23	3	0.0008776	Yes	Reject
PRRP	30	4	$1.41e^{-05}$	Yes	Reject
CLUSTALX	31	3	$9.276e^{-07}$	Yes	Reject
SAGA (with RBT-GA test sets)	31	3	$2.81e^{-06}$	Yes	Reject
DIALIGN	32	2	$5.006e^{-07}$	Yes	Reject
HMMT	34	0	$3.828e^{-07}$	Yes	Reject
SB_PIMA	34	0	$3.828e^{-07}$	Yes	Reject
ML_PIMA	33	1	$7.133e^{-07}$	Yes	Reject
MULTALIGN	32	2	$5.472e^{-07}$	Yes	Reject
PILEUP8	33	0	$5.654e^{-07}$	Yes	Reject
RBTGA	28	6	$6.918e^{-06}$	Yes	Reject

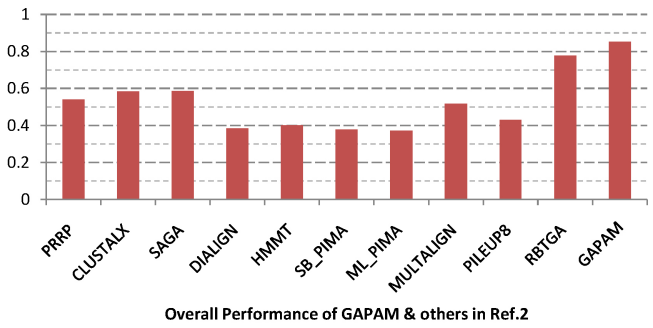

Fig. 14. Overall performance of all methods in reference 2 datasets.

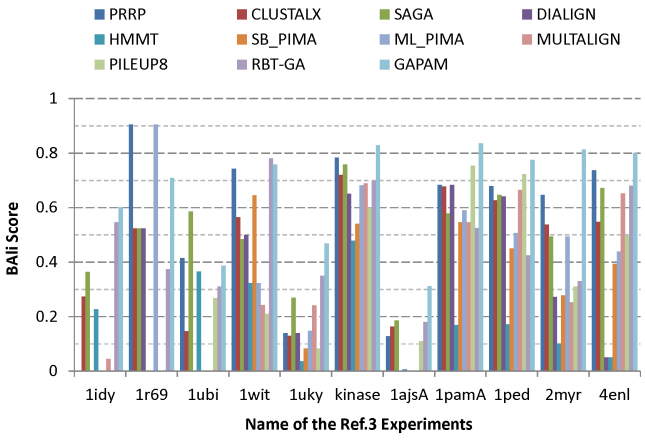

Fig. 15. Graphical presentations of the experimental results on reference 3 datasets.

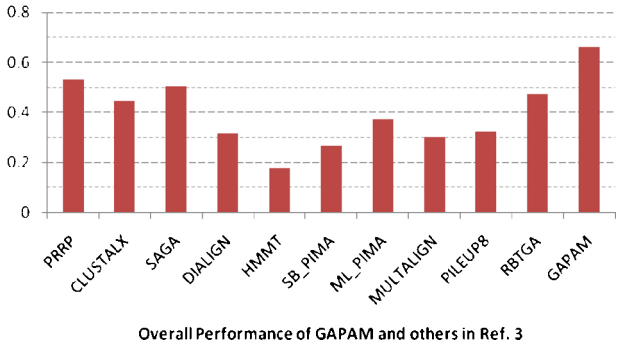

Fig. 16. Overall performance of all methods in reference 3 datasets.

more accurate MSAs in eight test cases, RBT-GA in 1, SAGA in 1, PRRP in 1, and ML_PIMA in 1 test case. PRRP and ML_PIMA found the same solution for one test case (1r69). Fig. 15 shows that for some test cases, most of the methods could not find any similarities in their solutions in comparison to the reference alignments. Therefore, these methods received zero score. PRRP and ML_PIMA achieved the same highest score in one test case, but both received a zero score for another test case. However, GAPAM did not obtain any zero score.

The overall performance of all methods for this reference is presented in Fig. 16. Although the GAPAM method did not achieve high accuracy solutions in three test cases, the average performance of this method was clearly better than the others as shown in Fig. 16.

c) *Statistical analysis:* To study the difference between any two stochastic algorithms in a more meaningful way, we have performed statistical significant testing. We have chosen a non-parametric test, Wilcoxon signed rank test [59] as it allows us to judge the difference between paired scores when it cannot make the assumption required by the paired-samples *t* test, such as that the population should be normally distributed. The results based on the best found solutions of GAPAM are presented in Table VII, where W (= W_+ or W_-) is the sum of ranks based on the absolute value of the difference between two test variables. The sign of the difference between two independent samples is used to classify cases into one of two samples: differences below zero (negative rank W_-), or above zero (positive rank W_+). As a null hypothesis, it is assumed

that there is no significant difference between two samples. Hence if the hypothesis test rejects the null hypothesis, then there is a significant difference, but if it retains the null hypothesis then there is no significant difference. The alternative hypothesis is that there is a significant difference in the fitness values of the two samples. The number of test problems is $N = 26$ and 34 for MSA-GA and RBT-GA respectively, and we used the 5% significance level. Based on the test results/rankings, we assigned two words ("yes" for $P <= 0.05$ or "no" for $P > 0.05$) for the comparison of any two algorithms (as shown in the fifth column), where "yes" means that the GAPAM algorithm is significantly better than the second, and "no" means that there is no significant difference between the two algorithms. We tested for significant with the BAliscore corresponding to the best found WSPM scores produced by GAPAM, in comparison to the published BAliscore results of the other methods

There is a significant difference when GAPAM is compared with MSA-GA, MSA-GA w/prealign and CLUSTAL W for the dataset used in MSA-GA, and when compared GAPAM with PRRP, CLUSTALX, DIALIGN, HMMT, SB_PIMA, ML_PIMA, MULTALIGN, PILEUP8 and RBT-GA for the dataset used in RBT-GA. In one case, however, there was no significant difference with SAGA for the dataset used in MSA-GA, as indicated by the hypothesis test decision and the significance values in Table VII. From the experimental observation, it is clear that GAPAM is statistically significantly better according to the Wilcoxon signed rank test.

VI. Conclusion

In this paper, a new GA based progressive alignment method (GAPAM) has been proposed to solve multiple sequence alignment problems. This approach works with the solution of a guide tree. To generate an initial population, two mechanisms are introduced. To assess the good performance of the algorithm, a number of the experiments tested the initial population, the genetic operators and the choice of an appropriate set of parameters for the GA. An initial experiment was run to determine the parameters, and from the experimental results, the probability of crossover and mutation was set to 50%–50%. A simple hill climbing method with the standard GAPAM initial population was tested to verify the performance of the genetic operators. Moreover, the GAPAM method was also tested with randomly generated initial populations to verify the performance of the proposed initial generation.

We used both the sum of pair and the weighted sum of pair methods for the fitness function in our research. However, we only reported the weighted sum of pair scores as the algorithm with this method performed better than the sum of pair fitness function.

To test our proposed approach, we considered a good number of benchmark datasets from BAliBase 2.0, so as to cover all the test sets of MSA-GA and RBT-GA. The proposed method was optimized based on the weighted sum of pair score. Therefore, the corresponding BAliscore of this solution was used to compare with other methods, as they used BAliscore as their measure of the quality/accuracy of the multiple sequence alignments. The experimental results showed that GAPAM performed better for most of the test cases. Although the solution of GAPAM was not always the best for some test cases, it was always close to the best. The overall performance of our proposed method outperformed all of the other methods considered in this paper. The GAPAM method performed better than the others because of its proposed initial generation, genetic operators and parameters.

After the statistical analysis and experimental analysis, we can safely conclude that the proposed method can effectively solve multiple sequence alignment problems.

References

[1] M. Höchsmann, B. Voss, and R. Giegerich, "Pure multiple RNA secondary structure alignments: A progressive profile approach," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 1, no. 1, pp. 53–62, Jan.–Mar. 2004.

[2] C. Notredame, "Recent progresses in multiple sequence alignment: A survey," *Pharmacogenomics*, vol. 3, no. 1, pp. 131–144, 2002.

[3] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, Mar. 1970.

[4] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, Mar. 1981.

[5] 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 Res.*, vol. 22, no. 22, pp. 4673–4680, Nov. 1994.

[6] J. D. Thompson, F. Plewniak, and O. Poch, "A comprehensive comparison of multiple sequence alignment programs," *Nucleic Acids Res.*, vol. 27, no. 13, pp. 2682–2690, 1999.

[7] J. Stoye, S. W. Perrey, and A. W. M. Dress, "Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment," *Appl. Math. Lett.*, vol. 10, no. 2, pp. 67–73, Mar. 1997.

[8] P. Bonizzoni and G. D. Vedova, "The complexity of multiple sequence alignment with SP-score that is a metric," *Theor. Comp. Sci.*, vol. 259, nos. 1–2, pp. 63–79, May 2001.

[9] D. F. Feng and R. F. Dolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *J. Mol. Evol.*, vol. 25, no. 4, pp. 351–360, 1987.

[10] O. Gotoh, "Optimal alignment between groups of sequences and its application to multiple sequence alignment," *Comput. Appl. Biosci.*, vol. 9, no. 3, pp. 361–370, Jun. 1993.

[11] A. V. Lukashin, J. Engelbrecht, and S. Brunak, "Multiple alignment using simulated annealing: Branch point definition in human mRNA splicing," *Nucleic Acids Res.*, vol. 20, no. 10, pp. 2511–2516, May 1992.

[12] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wooton, "Detecting subtle sequence signals: q Gibbs sampling strategy for multiple alignment," *Science*, vol. 262, pp. 208–214, Oct. 1993.

[13] R. F. Smith and T. F. Smith, "Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modeling," *Protein Eng.*, vol. 5, no. 1, pp. 35–41, 1992.

[14] B. Morgenstern, A. Dress, and T. Werner, "Multiple DNA and protein sequence alignment based on segment-to-segment comparison," *Proc. Natl. Acad. Sci. USA*, vol. 93, no. 22, pp. 12098–12103, Oct. 1996.

[15] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *J. Mol. Biol.*, vol. 302, no. 1, pp. 205–217, Sep. 2000.

[16] N. Bray, I. Dubchak, and L. Pachter, "AVID: A global alignment program," *Genome Res.*, vol. 13, no. 1, pp. 97–102, Jan. 2003.

[17] S. Schwartz, Z. Zhang, K. A. Frazer, A. Smit, C. Riemer, J. Bouck, R. Gibbs, R. Hardison, and W. Miller, "PipMaker: A web server for aligning two genomic DNA sequences," *Genome Res.*, vol. 10, no. 4, pp. 577–586, Apr. 2000.

[18] A. L. Delcher, S. Kasif, R. Leischman, J. Peterson, O. White, and S. L. Salzberg, " Alignment of whole genomes," *Nucleic Acids Res.*, vol. 27, no. 11, pp. 2369–2376, Jun. 1999.

[19] M. Brudno, C. B. Do, G. M. Cooper, M. F. Kim, E. Davydov, NISC Comparative Sequencing Program, E. D. Green, A. Sidow, and S. Batzoglou, "LAGAN and multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA," *Genome Res.*, vol. 13, no. 4, pp. 721–731, 2003.
[20] G. J. Barton and M. J. E. Sternberg, "A strategy for the rapid multiple alignment of protein sequences," *J. Mol. Biol.*, vol. 198, no. 2, pp. 327–337, Nov. 1987.
[21] W. Taylor, "A flexible method to align large numbers of biological sequences," *J. Mol. Biol.*, vol. 28, nos. 1–2, pp. 161–169, 1988.
[22] J. Devereux, P. Haeberli, and O. Smithies, "A comprehensive set of sequence analysis programs for the VAX," *Nucleic Acids Res.*, vol. 12, pp. 387–395, Jan. 1984.
[23] J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins, "The CLUSTAL_X windows interface: Flexible strategies for multiple sequence alignment aided by quality analysis tools," *Nucleic Acids Res.*, vol. 25, no. 24, pp. 4876–4882, Dec. 1997.
[24] P. H. A. Sneath and R. R. Sokal, "Taxonomic structure," in *Numerical Taxonomy*. San Francisco, CA: Freeman, 1973, pp. 188–308.
[25] N. Saitou and M. Nei, "The neighbor-joining method: A new method for reconstructing phylogenetic tree," *J. Mol. Biol.*, vol. 4, no. 4, pp. 406–425, Jul. 1987.
[26] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, no. 5, pp. 1792–1797, Mar. 2004.
[27] K. Katoh, K. Kuma, H. Toh, and T. Miyata, "MAFFT version 5: Improvement in accuracy of multiple sequence alignment," *Nucleic Acids Res.*, vol. 33, no. 2, pp. 511–518, 2005.
[28] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic consistency-based multiple sequence alignment," *Genome Res.*, vol. 15, no. 2, pp. 330–340, Feb. 2005.
[29] O. Gotoh, "Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments," *J. Mol. Biol.*, vol. 264, no. 4, pp. 823–838, 1996.
[30] J. Kim, S. Pramanik, and M. J. Chung, "Multiple sequence alignment using simulated annealing," *Comput. Applicat. Biosci.*, vol. 10, no. 4, pp. 419–426, 1994.
[31] A. V. Lukashin, J. Engelbrecht, and S. Brunak, "Multiple alignment using simulated annealing: Branch point definition in human mRNA splicing," *Nucleic Acids Res.*, vol. 20, no. 10, pp. 2511–2516, 1992.
[32] L. Cai, D. Juedes, and E. Liakhovitch, "Evolutionary computation techniques for multiple sequence alignment," in *Proc. CEC*, 2000, pp. 829–835.
[33] K. Chellapilla and G. B. Fogel, "Multiple sequence alignment using evolutionary programming," in *Proc. CEC*, 1999, pp. 445–452.
[34] T. T. Horng, C. M. Lin, B. J. Liu, and C. Y. Kao, "Using genetic algorithms to solve multiple sequence alignments," in *Proc. GECCO*, 2000, pp. 883–890.
[35] M. Ishikawa, T. Toya, Y. Totoki, and A. Konagaya, "Parallel iterative aligner with genetic algorithm," in *Proc. 4th Genome Informatics Workshop*, 1993, pp. 13–22.
[36] C. Notredame and D. G. Higgins, "SAGA: Sequence alignment by genetic algorithm," *Nucleic Acids Res.*, vol. 24, no. 8, pp. 1515–1524, 1996.
[37] S. R. Eddy, "Multiple alignment using hidden Markov models," *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, vol. 3, pp. 114–120, Jul. 1995.
[38] Z. J. Lee, S. F. Su, C. C. Chuang, and K. H. Liu, "Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 55–78, Jan. 2008.
[39] R. Thomsen, G. B. Fogel, and T. Krink, "A clustal alignment improver using evolutionary algorithms," in *Proc. CEC*, vol. 1. 2002, pp. 121–126.
[40] D. G. Higgins, A. J. Bleasby, and R. Fuchs, "CLUSTAL V: Improved software for multiple sequence alignment," *Bioinformatics*, vol. 8, no. 2, pp. 189–191, 1992.
[41] C. Shyu, L. Sheneman, and J. A. Foster, "Multiple sequence alignment with evolutionary computation," *Genet. Programming Evol. Mech.*, vol. 5, no. 2, pp. 121–144, Jun. 2004.
[42] C. Gondro and B. P. Kinghorn, "A simple genetic algorithm for multiple sequence alignment," *Genet. Mol. Res.*, vol. 6, no. 4, pp. 964–982, 2007.
[43] J. Taheri and A. Y. Zomaya, "RBT-GA: A novel metaheuristic for solving the multiple sequence alignment problem," *BMC Genomics*, vol. 10, no. 1:S10, pp. 1–11, Jul. 2009.
[44] A. Bahr, J. D. Thompson, J.-C. Thierry, and O. Poch, "BALIBASE (benchmark alignment dataBASE): Enhancements for repeats, transmembrane sequences and circular permutation," *Nucleic Acids Res.*, vol. 29, no. 1, pp. 323–326, 2000.
[45] J. Taheri, A. Y. Zomaya, and B. B. Zhou, "RBT-I: A novel approach for solving the multiple sequence alignment problem," in *Proc. 6th ACS/IEEE AICCSA*, 2008, pp. 28–36.
[46] J. Taheri, A. Y. Zomaya, and B. B. Zhou, "RBT-L: A location based approach for solving the multiple sequence alignment problem," *Int. J. Bioinformatics Res. Applicat.*, vol 6, no. 1, pp. 37–57, Jan. 2010.
[47] E. Zhang and A. K. C. Wong, "A genetic algorithm for multiple molecular sequence alignment," *Comput. Applicat. Biosci.*, vol. 13, no. 6, pp. 565–581, 1997.
[48] L. Jiao and L. Wong, "Novel genetic algorithm based on immunity," *IEEE Trans. Syst. Man Cybern. A*, vol. 30, no. 5, pp. 552–561, Sep. 2000.
[49] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989.
[50] A. Kolen and E. Pesch, "Genetic local search in combinatorial optimization," *Discr. Appl. Math. Combin. Oper. Res. Comput. Sci.*, vol. 48, no. 3, pp. 273–284, 1994.
[51] N. L. J. Ulder, E. H. L. Aarts, H. J. Bandelt, P. J. M. Van Laarhoven, and E. Pesch, "Genetic local search algorithms for the traveling salesman problem," in *Proc. 1st Workshop PPSN*, vol. 496. 1991, pp. 109–116.
[52] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, "A model of evolutionary change in proteins," *Atlas Protein Sequence Structure*, vol. 5, no. 3, pp. 345–351, 1978.
[53] S. Vinga and J. Almeida, "Align-free sequence comparison: A review," *Bioinformatics*, vol. 19, no. 4, pp. 513–523, 2003.
[54] M. Kimura, "A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences," *J. Mol. Evol.*, vol. 16, no. 2, pp. 111–120, Dec. 1980.
[55] K. Yang and L. Zhang, "Performance comparison between *k*-tuple distance and four model-based distances in phylogenetic tree reconstruction," *Nucleic Acids Res.*, vol. 36, no. 5, p. e33, Mar. 2008.
[56] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Natl. Acad. Sci. USA*, vol. 89, no. 22, pp. 10915–10919, Nov. 1992.
[57] *GCG Manual* [Online]. Available: http://www.umdnj.edu/gcg/gcgmanual.html
[58] J. D. Thompson, F. Plewniak, and O. Poch, "BAliBASE: A benchmark alignments database for the evaluation of multiple sequence alignment programs," *Bioinformatics*, vol. 15, no. 1, pp. 87–88, Jan. 1999.
[59] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New York: Wiley, 2009.

Farhana Naznin (M'11) received the B.Sc. (honors) degree in electrical and electronic engineering from the Rajshahi University of Engineering and Technology, Rajshahi, Bangladesh, in 2003, and the M.Sc. degree in information engineering from the University of the Ryukyus, Nishihara, Okinawa, Japan, in 2007. She is currently pursuing the Ph.D. degree in computer science with the School of Engineering and Information Technology, University of New South Wales at Australian Defense Force Academy, Canberra, Australia.

Her main research interests include evolutionary optimization in bioinformatics.

Ruhul Sarker (M'03) received the Ph.D. degree in operations research from DalTech (formerly TUNS), Dalhousie University, Halifax, NS, Canada, in 1991.

He is currently an Associate Professor and the Deputy Head of School (Research) with the School of Engineering and Information Technology, University of New South Wales at Australian Defense Force Academy, Canberra, Australia. He has published more than 190 refereed papers in journals, books and conference proceedings, and has edited 8 books on specialized topics. He is the lead author of the book *Optimization Modelling: A Practical Approach* (Taylor and Francis, 2008). His main research interests include applied operations research and evolutionary optimization.

Dr. Sarker was a Technical Co-Chair of 2003 IEEE CEC and is a Publications Co-Chair for WCCI 2012.

Daryl Essam received the B.S. degree from the University of New England, Armidale, NSW, Australia, in 1990, and the Ph.D. degree from the University of New South Wales, Kensington, NSW, Australia, in 2000.

He was an Associate Lecturer with the University of New England in 1991 and has been with the School of Engineering and Information Technology, University of New South Wales at Australian Defense Force Academy, Canberra, Australia, since 1994, where he is currently a Senior Lecturer. His current research interests include genetic algorithms with a focus on both genetic programming and multiobjective optimization.