COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE 114 (2014) 38–49

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

Multiple sequence alignment with affine gap by using multi-objective genetic algorithm

Mehmet Kaya[a], *Abdullah Sarhan*[b], *Reda Alhajj*[b,c,*]

[a] *Department of Computer Engineering, Firat University, 23119 Elazig, Turkey*
[b] *Department of Computer Science, University of Calgary, Calgary, AB, Canada*
[c] *Department of Computer Science, Global University, Beirut, Lebanon*

ARTICLE INFO

Article history:
Received 20 April 2013
Received in revised form
29 November 2013
Accepted 12 January 2014

Keywords:
Multiple sequence alignment
Multi-objective genetic algorithm
Bioinformatics

ABSTRACT

Multiple sequence alignment is of central importance to bioinformatics and computational biology. Although a large number of algorithms for computing a multiple sequence alignment have been designed, the efficient computation of highly accurate and statistically significant multiple alignments is still a challenge. In this paper, we propose an efficient method by using multi-objective genetic algorithm (MSAGMOGA) to discover optimal alignments with affine gap in multiple sequence data. The main advantage of our approach is that a large number of tradeoff (i.e., non-dominated) alignments can be obtained by a single run with respect to conflicting objectives: affine gap penalty minimization and similarity and support maximization. To the best of our knowledge, this is the first effort with three objectives in this direction. The proposed method can be applied to any data set with a sequential character. Furthermore, it allows any choice of similarity measures for finding alignments. By analyzing the obtained optimal alignments, the decision maker can understand the tradeoff between the objectives. We compared our method with the three well-known multiple sequence alignment methods, MUSCLE, SAGA and MSA-GA. As the first of them is a progressive method, and the other two are based on evolutionary algorithms. Experiments on the BAliBASE 2.0 database were conducted and the results confirm that MSAGMOGA obtains the results with better accuracy statistical significance compared with the three well-known methods in aligning multiple sequence alignment with affine gap. The proposed method also finds solutions faster than the other evolutionary approaches mentioned above.

1. Introduction

The alignment of DNA, RNA or protein sequences is a recurring task in computational biology. Aligning a set of sequences allows for identifying the regions where the sequences are similar and where they differ [1]. There are some studies tackling the computational complexity of two popular problems in multiple sequence alignment; multiple alignment with sum-of-pairs score and multiple tree alignment [2]. For instance, Bonizzoni and Vedova analyzed the computational complexity of computing the optimal alignment of a set of sequences under the sum-of-pairs score scheme [3]. It is already known that Multiple Sequence Alignment (MSA) problems with sum-of-pairs score can be solved in n^k steps for k sequences of length at most n via dynamic programming [4]. Although dynamic programming guarantees optimal alignment mathematically, it can only handle MSA problems with a small number of short sequences. However, the size of the MSA problem space rises dramatically with the number of sequences

* *Corresponding author*. Tel.: +1 403 210 9453.
E-mail addresses: kaya@firat.edu.tr (M. Kaya), sarhanabed@gmail.com (A. Sarhan), alhajj@ucalgary.ca, rsalhajj@gmail.com (R. Alhajj).
0169-2607/$ – see front matter © 2014 Elsevier Ireland Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cmpb.2014.01.013

and their lengths. In such a case, the MSA problem with sum-of-pairs score is NP-hard.

In order to align multiple sequences within limited resources, two main approaches have been proposed. The progressive approach is a greedy heuristic algorithm [5]. It involves estimating a guide tree from unaligned sequences, and then incorporating the sequences into the alignment with a pairwise alignment algorithm. Several variations of this basic strategy have been developed [6–8]. From those, ClustalW is a global method for multiple sequence algorithms, which improves the local optimality issue of the progressive approach [8]. Although this approach is successful in a wide variety of cases, this method suffers from its greediness [9].

In order to overcome the limitations of the progressive alignment methods, recently, the trend has shifted to the use of iterative algorithms with the progressive approach, such as MUSCLE [10], MAFFT [11] and ProbCons [12], and with the stochastic approach, such as simulated annealing [13], genetic algorithms [14], hidden Markov model [15] and Gibbs Sampling [16]. Iterative approaches seek to optimize a scoring function that reflects biological events such that optimization of the score leads to a correct alignment [17]. MUSCLE [10] is a progressive and iterative algorithm which includes three stages; draft progressive, improve progressive and refinement. In each stage, a multiple sequence alignment is generated. Similarly, MAFFT [11] is also based on a progressive and iterative algorithm. It employs a Fast Fourier Transform to identify homologous regions. ProbCons [12] is a Probabilistic and Consistency based algorithm. It obtains more accurate results than MUSCLE and MAFFT, but it is slower than these algorithms.

As an example of iterative and stochastic methods, genetic algorithms, such as SAGA [18], MSA-GA [19], RBT-GA [20] and GAPAM [22] have been successfully employed to find good alignments. In SAGA, the initial generation is created randomly. For this method, 22 different operators are used to gradually improve the fitness of MSA. These operators are dynamically scheduled during the evolution process. The time complexity of SAGA is large because of the time required by the repeated use of the fitness function. MSA-GA is a simple genetic algorithm based method using a different scoring function. In order to test this algorithm, the authors performed two sets of five runs for each of 28 test cases from the BAliBASE 2.0 dataset [21]. RBT-GA is also a GA based method, combined with the Rubber Band Technique (RBT), to find optimal protein sequence alignments. Finally, GAPAM is a progressive alignment method using a genetic algorithm for multiple sequence alignment [22]. It introduces two 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. The experimental results show that GAPAM obtained good results.

However, all of the studies above related to genetic algorithms employ a single-objective to align multiple sequences, although different methods of fitness calculation are used. Recently, a few works have been implemented on multi-objective genetic based sequence alignment. For instance, Seeluangsawat and Chongstitvatana proposed a method called MOMSA. The method solves the alignment problem by using two objective functions. The objective functions consider the largeness of the gap opening penalty and the gap extension penalty. [23] As another approach, Taneda handles secondary structure score and sequence similarity score for multi-objectives [24]. Parallel Niche Pareto AlineaGA uses two objectives for performing multiple sequence alignment. One of them is the sum of pairs and the other is the number of fully identical columns in the alignment [25]. Ortuno et al. implemented a multi-objective evolutionary algorithm based on NSGA II in order to assemble previously aligned sequences, trying to avoid suboptimal alignments [26]. Finally, DeRonne and Karypis generated pareto optimal pairwise sequence alignments for all combinations of two, three and four objectives. The eleven objectives used in their study are the functions proposed by some other authors [27]. Also, all of the methods try to align all of the given sequences. In other words, although one or several sequences are not very similar to the rest of the sequences in the alignment, they are forced to align with those sequences. If these non-similar sequences align together with the others, the similarity value used to compare sequences decreases. In this paper, to address all the problems listed and mentioned above, we propose a multi-objective GA based method for multiple sequence alignment. The paper demonstrates advantages of the multi-objective approach over single-objective ones to align multiple sequences efficiently and effectively. For this purpose, we use the following three-objective based formulation to find the best possible solutions with smaller alignment length:

> Maximize *similarity*, minimize *affine gap penalty*, and maximize *support*

The meanings of these objectives will be discussed in the objectives subsection in detail.

We performed experiments on BAliBASE 2.0 to demonstrate the effectiveness of our method. The experimental results show the superiority of the proposed method, in terms of affine gap penalty, accuracy, statistical significance and the runtime required to find optimal alignment, over three well known multiple sequence alignment methods, MUSCLE, SAGA and MSA-GA.

The rest of the paper is organized as follows. Section 2 presents some preliminary concepts regarding multiple sequence alignment and multi-objective optimization. Section 3 introduces the proposed method. Section 4 reports our findings obtained from the application of the proposed approach to BAliBASE 2.0. Finally, summary, conclusions and the future work are included in Section 5.

2. Preliminaries

2.1. *Multiple sequence alignment*

Given a finite alphabet set $\sum$ and a family $S = (s_1, s_2, \ldots, s_k)$ of k sequences of varying length l_1 to l_k:

$$s_i = s_{i1}s_{i2}, \ldots, s_{il_i} (1 \le i \le k), \quad s_{ij} \in \sum (1 \le j \le l_i) \tag{1}$$

where for DNA sequences, $\sum$ consists of 4 characters of the nucleotides {A, T, G, C} and for protein sequences, $\sum$ consists

of 20 characters of the amino acids; an alignment of S is a matrix

$$A = (a_{ij})_{1\le i\le k, 1\le j\le l, \max(l_i)\le l\le \sum_{i=1}^{k} l_i} \quad (2)$$

satisfying:

(1) $a_{ij} \in \sum \cup \{-\}$, here "–" denotes the gap letters;
(2) each row $a_i = a_{i1}a_{i2}, \ldots, a_{il}$ $(1 \le i \le n)$ of A is exactly the corresponding sequence s_i if we eliminate all the gap letters;
(3) A has no column which only contains gaps.

2.2. Multi-objective optimization

Many real world problems involve multiple measures of performance or objectives, which should be optimized simultaneously. Multi-objective optimization (MOO) operates by seeking to optimize the component of a vector-valued objective function. Unlike single-objective optimization, the solution to a MOO problem is a family of points known as the Pareto-optimal set. Each point in the set is optimal in the sense that no improvement can be achieved in one component of the objective vector that does not lead to degradation in at least one of the remaining components. Given a set of possible solutions, a candidate is said to be Pareto-optimal if there are no other solutions in the solution set that can dominate any of the candidate solutions. In other words, the candidate solution would be a non-dominated solution. A general minimization problem of M objectives can be mathematically stated as

$$\text{Minimize } f(x) = [f_i(x), i = 1, \ldots, M] \quad (3)$$

subject to:

$$g_j(x) \le 0 \quad j = 1, 2, \ldots, J$$

$$h_k(h) \le 0 \quad k = 1, 2, \ldots, K$$

where f_i, g_j and h_k are the ith objective function, jth inequality and kth equality constraints, respectively, x is a decision vector that represents a solution, M, J and K are the number of objectives, inequality and equality constraints.

In general, the objectives of the optimization problem are often conflicting. Optimal performance according to one objective, if such an optimum exists, often implies unacceptable low performance in one or more of the other objective dimensions, creating the need for a compromise to be reached. A suitable solution to such problems involving conflicting objectives should offer *acceptable*, though possibly sub-optimal in the single objective sense, performance in all objective dimensions, where acceptable is a problem dependent and ultimately subjective concept. An important concept in MOO is that of domination, where a solution x_i is said to dominate another solution x_j if the following two conditions are true:

- the solution x_i is not worse than x_j in all objectives;
- the solution x_i is strictly better than x_j in at least one objective.

This, in turn, leads to the definition of *Pareto-optimality*, where a decision vector $x_i \in U$, where U stands for the universe, is said to be *Pareto-optimal* if and only if there exists no x_j, $x_j \in U$, such that x_i is dominated by x_j. Solution x_i is said to be *non-dominated*. The set of all such non-dominated solutions is called the *Pareto-optimal set* or the *non-dominated set*. In general, MOO problems tend to achieve a family of alternatives which must consider the relevance of each objective relative to the other objectives [28]. Recently, some researchers have studied different problems by using multi-objective genetic algorithms. We have already participated in some of these efforts within the data mining domain [29,30].

3. The MSAGMOGA algorithm

We use a well-known high-performance multi-objective genetic algorithm called NSGA II [31] to find the best possible alignments with affine gap from multiple sequences with respect to three objectives, which will be discussed in the objectives subsection. We first give the following background information about the process because our algorithm is based on GAs.

A GA is a search and optimization methodology from the field of evolutionary computation that was invented by Holland [32]. A GA is based on the Darwin's natural selection principle of the survival of the fittest, and is widely used for hard problems in engineering and computer science. A GA is a population-based method where each individual of the population represents a candidate solution for the target problem. This population of solutions evolves throughout several generations, starting from a randomly generated one, in general. During each generation of the evolutionary process, each individual of the population is evaluated by a fitness function, which measures how good the solution represented by the individual is for the target problem. From a given generation to another, some parent individuals (usually those having the highest fitness) produce "offsprings", i.e., new individuals that inherit some features from their parents, whereas others (with low fitness) are discarded, following Darwin's principle of natural selection. The selection of the parents is based on a probabilistic process, biased by their fitness value. Following this procedure, it is expected that, on average, the fitness of the population will not decrease every consecutive generation. The generation of new offsprings, from the selected parents of the current generation, is accomplished by means of genetic operators. This process is iteratively repeated until a satisfactory solution is found or some stop criterion is reached, such as the maximum number of generations.

3.1. Structure of the individuals

The population is initially randomly generated by loading each sequence to each gene of the individual, determining the length of the largest sequence, and completing each one of the sequences with gaps until they reach the length of the largest sequence plus a random number of gaps between 0 and 20% of the length of the largest loaded sequence. These gaps are randomly positioned into sequences. The choice of 20% is based on the observation that solutions to the multiple

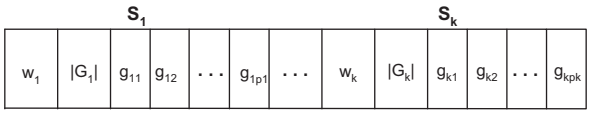

Fig. 1 – Representation of an individual.

sequence alignment problems rarely contain more than 20% gaps [33].

An individual in our method mainly holds the positions of gaps on all the target sequences. In this meaning, an individual is divided into k genes, as shown in Fig. 1, where each gene is defined based on whether the corresponding sequence is present or not, number of the gaps and the positions of gaps. Fig. 1 shows the representation of the alignment with k x_i l_i to l_k. The genes are positional, i.e., the first gene deals with the first sequence, the second gene deals with the second sequence, and so on.

The field *weight* (w_i) is a real-valued variable normalized into the interval [0,1]. This variable indicates whether or not the corresponding sequence will be considered. More precisely, when w_i is smaller than a user-defined threshold (called *Limit*) the corresponding sequence will not be considered in the alignment. Therefore, the greater the value of the threshold *Limit*, the smaller is the probability that the corresponding sequence will be included in aligning multiple sequences.

The field $|G_i|$ was used to determine the number of gaps in the ith sequence ($i = 1,2, \ldots, k$). The value of this part changes between $(\max(l_m) - l_i)$ and $(\max(l_m) + [20\%.\max(l_m)] - l_i)$, $(m = 1,2, \ldots, k)$, where (l_m) is the length of the largest sequence, l_i is the length of the ith sequence.

The fields $g_{i1}, g_{i2}, \ldots, g_{ip_1}$ indicate the positions of the gaps in the ith sequence, where $0 \leq g_{ij} \leq l_i$ and

$$p_i = ((\max(l_m) + \lceil 20\%.\max(l_m) \rceil) - l_i) \tag{4}$$

$g_{ij} = 3$ means there is a gap after the third nucleotide. If two 3's exist in the g_{ij} fields regardless of the order, this shows that there are two gaps after the third nucleotide.

It should be noted that although the number of g_{ij} fields is p_i, $|G_i|$-number of those fields is considered. As an example, let us assume that we have three sequences for alignment. Their lengths are 10, 15, and 20. In this case, the length of the largest sequence plus 20% of this sequence is 24. Thus, the number of the gaps which the first sequence will have changes in the interval [10,14]; this range is [5,9] for the second sequence, and it is [0,4] for the third sequence.

Note that the above encoding is quite flexible with respect to the number of the sequences. A traditional GA is very limited in this aspect, since it can only cope with fixed-number sequences. In our approach, although each individual has a fixed length, the genes are interpreted (based on the value of the weight w_i) in such a way that the individual phenotype (the alignment) has a variable length. Hence, different individuals correspond to alignments with different lengths.

The start of the first population consists of generating, arbitrarily, a fixed number of individuals during the evolution.

3.2. *Objectives and selection*

The fitness of an individual in MSAGMOGA is assessed on the basis of *similarity*, *affine gap penalty* and *support*. All the objectives defined in this section are valid for both DNA/RNA and protein sequences.

3.2.1. *Similarity*

It performs a measure of similarity among all sequences defining an individual. To calculate this, we first generate a position weight matrix from the alignment found by MSAGMOGA. Then, the dominance value (dv) of the dominant nucleotide in each column is found as follows:

$$dv(i) = \max_b \{f(b, i)\}, \quad i = 1, \ldots, l \tag{5}$$

where $f(b,i)$ is the score of nucleotide b on column i in the position weight matrix regardless of the existence of a gap, $dv(i)$ is the dominance value of the dominant nucleotide on column i, and l is the alignment length.

We define the similarity objective function of alignment A as the average of the dominance values of all columns in the position weight matrix. In other words,

$$Similarity\,(A) = \frac{\sum_{i=1}^{l} dv(i)}{l} \tag{6}$$

The closer the value of the similarity (A) is to one, the larger the probability that the candidate alignment A will be discovered as a best possible alignment. To evaluate the score of a candidate alignment, the reward function and the penalty function are used together. The sum of pairs is the most widely used reward function. In this paper, we separated the reward function and the penalty function into two different objectives. As we try to maximize our similarity measure, at the same time, we aim to minimize the affine gap penalty. It should be noted that our similarity measure is different from the substitution matrices which provide a numerical score for matches and mismatches.

A substitution matrix is a table of numbers of dimension 20×20 for amino acids and 4×4 for nucleic acids which represents all possible transitions between the 20 amino acids or the 4 nucleic acids. They provide a measure of the probability of a substitution or conservation occurring. Since the direction of a substitution is unknown, the matrices are symmetric. For DNA sequences, a simple matrix commonly used assigns a positive value for a match and a negative value for a mismatch. For protein sequences the most common matrices are PAM [34] and BLOSUM [35]. 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 on only highly conserved regions in series of alignments forbidden to contain gaps.

In the proposed similarity method, we try to optimize the number of matches in the alignment regardless of evolutionary distance. So, we will maximize the numerical score for matches, using the strict ASCII code comparison. Of course, while doing this, the gap penalty and indirectly alignment length will increase. In order to overcome this problem,

Table 1 – The position weight matrix of an alignment with length 4.

A A C T
C A C T
T – T T
T A C T
– A C T

	1	2	3	4
A	0.25	1	0	0
C	0.25	0	0.8	0
T	0.5	0	0.2	1
G	0	0	0	0

Table 2 – The position weight matrix of an alignment with length 5.

F – F F D
P D F – D
P – F F D
P D D F D

	1	2	3	4	5
F	0.25	0	0.75	1	0
P	0.75	0	0	0	0
D	0	1	0.25	0	1

we present another objective function, called as affine gap penalty, in the next section.

In fact, our similarity measure does not directly depend on the affine gap penalty. That is, even if the affine gap penalty value is too large, the similarity value may be high. The following example shows how to compute the similarity measure in the given two position weight matrices with DNA and protein sequences:

Example. Consider two matrices, the first matrix (Table 1) implies that the number of target DNA sequences in a dataset is 5 and the alignment length is found to be 4. In such a case, the dominant character for column 1 is *T* and its dominance value is 0.5. The dominance values of the other columns are 1, 0.8 and 1, respectively. As for the similarity value, it is computed as:

$$\left(\frac{0.5+1+0.8+1}{4}\right) = 0.825$$

Similarly, in the second matrix (Table 2), the number of target protein sequences is 4 and the alignment length is determined to be 5. The similarity value is computed as 0.9 for this longer alignment.

3.2.2. *Affine gap penalty*

The affine gap cost model penalizes insertions and deletions using a linear function in which one term is length independent and the other is length dependent. The affine gap penalty in the pairwise alignment of rows *i* and *j* is determined by the following formula:

$$Gap_{ij}(p) = g_{open} + g_{extend}(p-1), \quad p > 1$$

where g_{open} is the gap opening penalty, g_{extend} is the gap extension penalty and p is the length of the gap string.

The following example clarifies this situation in more detail (Fig. 2).

If we assume $g_{open} = 5$ and $g_{extend} = 2$, as the affine gap penalty value of the first alignment (a) is $5+5+5+5=20$, and that of the second alignment (b) is $5+2+2+2=11$.

Although both of them have the same similarity value which is the objective that was proposed in this paper, the second one is often more plausible and more realistic in biological meaning. It is noted that a single insertion of "GAAT" into the second string could change it into the first.

In linear gap penalty, the cost of a run of p gaps is $gap\ x\ p$. It might be more realistic to support general gap penalty, so that the score of a run of k gaps is $(p) < gap\ x\ k$.

In this paper, our aim is to try to group gaps together. In other words, our objective is to minimize the following formula:

$$Affine\ gap\ penalty = \sum_{i=1}^{k-1}\sum_{j=1+1}^{k} Gap_{ij}(p)$$

3.2.3. *Support*

Here, the meaning of this objective is the same as that in the data mining field; that is, the frequency in the given sequences. Sometimes, if a sequence does not appear in the alignment, the alignment quality may be higher. In other words, if the sequences corrupting the alignment quality are removed, the alignment with the better quality can be found. However, in our method, since we try to maximize the support value as objective function, we can find pareto-optimal alternative solutions including all the sequences and/or leaving some sequences out by a single run at the same time. So, support is the number of sequences present in an alignment with quality. The greater the support value, the stronger is the alignment covered by most of the sequences in the dataset.

Overall, the multiple sequence alignment problem is converted into the following three-objective optimization problem:

Maximize similarity (A), *Minimize affine gap penalty* (A),

and maximize support (A)

The individuals in a population are first sorted based on their domination status using the procedure non-dominated sort [31]. Here, the individuals that are not dominated by any other member of the population form the first front, and are put in rank 1. These individuals are then removed from consideration, and the individuals which thereafter become nondominated are assigned rank 2. The process is repeated until all the individuals have been assigned a rank.

A A A G A A T T C A A A A G A A T T C A
A – A – A – T – C A A A A – – – – T C A

(a) **(b)**

Fig. 2 – Two different alignment cases.

The individuals are then put in a sorted order according to their ranks. The overall complexity of the nondominated sort algorithm described in detail in [31] is shown to be $O(M N^2)$, where M is the number of objectives and N is the number of individuals in the population. The first front represents a nondominated set with respect to the current population, since none of the solutions in this front is dominated by any other solution in the population.

After applying the nondominated sort algorithm, the individuals are assigned a crowding distance and the selection is performed using the crowded tournament selection. The crowding distance is computed as the sum of the difference of the objective values of the solutions preceding and following the current solution (corresponding to the individual under consideration) for each objective. This provides a measure of the density of the solution surrounding a particular point in the population.

In crowded tournament selection, a pair of individuals is selected randomly, and the one with the lower rank is selected. If the ranks of the individuals are equal, then the individual with a larger crowding distance is chosen. The large crowding distance ensures that the solutions are spread along the Pareto-optimal front.

3.3. *Genetic operators*

The principle of crossover is exchanging the information of chromosomes to produce offsprings. The chromosomes with better performance might be produced through preserving good structures of the parent chromosomes. The usual one-point crossover operator is stochastically applied with a predefined probability, using two individuals of the selected pool. The crossover point is a percentage of the length of the individual that defines the starting point from where the crossover breaks the string. We use two different types of crossover in MSAGMOGA: one-point and two-point. The two-point crossover combines two parent alignments through two exchanges.

The mutation operator is used to foster more exploration of the search space and to avoid unrecoverable loss of genetic material that leads to premature convergence to some local minima. In general, mutation is implemented by changing the value of a specific position of an individual with a given probability, denominated mutation probability. MSAGMOGA developed three mutation operators tailored for our genome representation:

Shift the gap toward the right: The value of g_{ij} of a randomly selected gene is increased by one.
Shift the gap toward the left: The value of g_{ij} of a randomly selected gene is decreased by one.
Random-changing: The mutation produces a small integer number that is then added to or subtracted from the current content of any weight, number of the gaps, or position of the gaps. This is implemented in such a way that the lower and upper bounds of the domain of the field are never exceeded.

To sum up, the MSAGMOGA process employed in this study can be summarized by the following algorithm,

The Algorithm.

Input: population size N; maximum number of generations G; crossover probability p_c; mutation rate p_m.
Output: nondominated set
Step 1: P: = Initialize (P)
Step2: *while* the termination criterion is not satisfied *do*
Step 3: C: = Select from (P)
Step 4: C^I: = Genetic operators (C)
Step 5: P: = Replace (PUC^I)
Step 6: *end while*
Step 7: return (P)

First, an initial population P is generated in Step 1. Pairs of parent solutions are chosen from the current population P in Step 3. The set of the selected pairs of the parent solutions is denoted by C in Step 3. The crossover and the mutation operations are applied to each pair in C to generate the offsprings population C^I in Step 4. The next population is constructed by choosing good solutions from the merged population PUC^I. The pareto-dominance relation and a crowding measure are used to evaluate each solution in the current population P in Step 3 and the merged population PUC^I in Step 5. Elitism is implemented in Step 5 by choosing good solutions as members in the next population from the merged solution PUC^I.

4. Experimental results

We conducted some experiments in order to analyze and demonstrate the efficiency and effectiveness of MSAGMOGA. Further, the superiority of the proposed approach has been demonstrated by comparing it with three multiple sequence alignment methods, namely MUSCLE, SAGA and MSA-GA. In our experiments, we concentrate on testing the ratio of the pairs correctly aligned (SP), the ratio of the columns correctly aligned (CS) and time requirements as well as changes in the main factors that affect the proposed multi-objective process, namely finding non-dominated sets, similarity, affine gap penalty and support. All of the experiments have been conducted on an Intel C2D 1.8 GHz CPU with 4GB of memory and running Windows 7. The proposed method was implemented in C++. As the data sets are concerned, we used seven different data sets with specific features chosen from BAliBASE 2.0. These features have been specifically chosen to test the runtime and accuracy of our method. As shown in Table 3, the sequence number of the second data set is larger as its sequence length is almost the same as that of the first data set. Similarly, the third data set has larger sequence length as the sequence number is the same as that of the first data.

In all our tests, the MSAGMOGA process started with a population of 200 individuals. As the termination criteria, if the best solution produced in each generation remains the same in 100 consecutive generations, the algorithm will be terminated. Finally, while the crossover probability is chosen to be 0.8, the mutation rate of 0.3 was used for each kind of mutation. Also, as the affine gap penalty is computed, g_{open} and g_{extend} were chosen as 5 and 2, respectively.

Table 3 – The data sets used in the experiments.

Data set	Identity	Sequence number	Shortest seq. length	Longest seq. length	Alignment length
1amk (Ref. [1])	>35%	5	242	254	258
2hsdA(Ref. [2])	–	22	225	262	291
1gpb (Ref. [1])	>35%	5	796	828	854
Kinase (Ref. [1])	<25%	5	260	273	316
1taq (Ref. [1])	>35%	5	806	928	950
1dynA (Ref. [4])	–	6	105	627	912
2myr (Ref. [3])	–	21	299	482	608

The computational complexity of MSAGMOGA is N.M.K. $[(\max(l_m) + 20\%.\max(l_m)) - \min(l_m)]$, $(m = 1,2, \ldots, k)$ where $\max(l_m)$ is the length of the longest sequence to be aligned, $\min(l_m)$ is the length of the shortest sequence, k is the number of sequences, N is the number of individuals in the population and M is the number of maximum generation. We tested MSAGMOGA up to 500 generations and we observed that the nondominated solutions were hardly changed after 250th generation. In other words, multi-objective values converged after this generation number.

Three sets of experiments for each data set were carried out. The first set of experiments is dedicated to evaluate the *1amk* data set. The data set contains 5 sequences. The length of the shortest sequence is 242, and the longest sequence is 254. In other words, the sequence lengths are almost the same. Furthermore, this data set has more than 35% of the identity among its sequences. Some non-dominated solutions found by MSAGMOGA are reported in Table 4. Here, the values of the affine gap penalty and similarity of some nondominated solutions are given for three different values of the support. As can be easily seen from Table 4, as the support value increases the affine gap penalty increases too. However, for each value of the support, as the affine gap penalty decreases, the similarity value decreases. Thus, the tradeoff between the similarity and the affine gap penalty is clearly observed for three values of the support.

In the second experiment of the first set, we showed the performance of our algorithm and the other methods. For this purpose, we have executed our algorithm for 10 runs and calculated the average scores. In comparison, SAGA and MSA-GA used 10 independent runs with the standard parameters defined in their papers. Then, we compared the results of our algorithm with the other methods, as shown in Table 5. It should be noted here that the our solution reported in Table 5 is the one having the values Support = 5 and Similarity = 0.532 given in Table 4. The bold face value represents the best score in Table 5. From this table, it can be observed that our method has obtained the best solution for the *1amk* data set.

Table 4 – The objective values of the non-dominated solutions for *1amk*.

Support	Affine gap penalty	Similarity
3	103	0.687
	106	0.696
	112	0.714
4	109	0.579
	118	0.588
	124	0.597
5	126	0.505
	129	0.516
	132	0.532

Table 5 – Test results of four different methods for *1amk*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	132	0.999	0.955
MUSCLE	134	0.991	0.942
SAGA	139	0.976	0.933
MSA-GA	143	0.965	0.927

The final experiment for this set compares the runtimes of four different methods. The results are reported in Fig. 3. The runtime reported for MSAGMOGA represents the non-dominated solution given in Table 5. As a result of this experiment, it has been observed that MSAGMOGA outperforms the other two well-known genetic algorithm based approaches, SAGA and MSA-GA, for the *1amk* data set. Fig. 4 shows the alignment result of the *1amk* based on our proposed method. From Fig. 4, it can be seen that MSAGMOGA can solve multiple sequence alignment problem successfully.

In the second set of the experiments, we applied MSAGMOGA on a data set having larger number of sequences, *2hsdA*, as its sequence length is the same as the previous data set. This new data set contains 22 sequences. The first experiment obtains the values of the objectives of the nondominated set for the *2hsdA* data set. Table 6 reports the results. As can be easily seen from Table 6, for two solutions having the same affine gap penalty (256 marked with * in Table 6) the similarity value (0.397) of the solution whose support value is higher (i.e., 22), is lower than that of the other.

The second experiment deals with judging the performance of our method with respect to the affine gap penalty, SP and CS. As in the previous experiment set, we have compared our results with MUSCLE, SAGA and MSA-GA. Our method has successfully found more accurate solution compared to the other methods for the *2hsdA* data set as shown in Table 7.

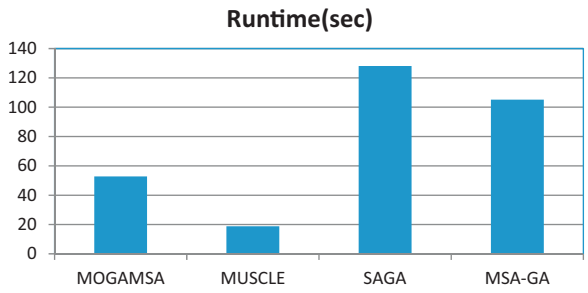

Fig. 3 – Comparison of the runtimes of four different methods for *1amk*.

```
Seq 1  .SAKPQPIAAANWKCNGTTASIEKLVQVFNEHTİS.HDVQCVVAPTFVHIPLVQAKLRNPKYVISAENAİAK.SGAFTGEVSMPILKDIGVHWVILGHSE
Seq 2  A..RT.FFVGGNFKLNGSKQSIKEIVERLNTASİP.ENVEVVICPPATYLDYSVSLVKKPQVTVGAQNAYLKASGAFTGENSVDQIKDVGAKWVILGHSE
Seq 3  APSRK.FFVGGNWKMNGRKQSLGELIGTLNAAKVP.ADTEVVCAPPTAYIDFARQKLD.PKIAVAAQNCYKVTNGAFTGEİSPGMIKDCGATWVVLGHSE
Seq 4  ..SKPQPIAAANWKCNGSQQSLSELIDLFNSTSİN.HDVQCVVASTFVHLAMTKERLSHPKFVIAAQNAGN........ADALASLKDFGVNWIVLGHSE
Seq 5  ..MRH.PLVMGNWKLNGSRHMVHELVSNLRKELAGVAGCAVAIAPPEMYIDMAKREAEGSHIMLGAQNVDLNLSGAFTGETSAAMLKDIGAQYIIIGHSE

Seq 1  RRTYYGETDEIVAQKVSEACKQGFMVIACIGETLQQREANQTAKVVLSQTSAIAAKLTKDAWNQVVLAYEPVWAIGTGKVATPEQAQEVHLLLRKWVSEN
Seq 2  RRSYFHEDDKFIADKTKFALGQGVGVILCIGETLEEKKAGKTLDVVERQLNAVLEEVK..DWTNVVVAYEPVWAIGTGLAATPEDAQDIHASIRKFLASK
Seq 3  RRHVFGESDELIGQKVAHALAEGLGVIACIGEKLDEREAGITEKVVFEQTKVIADNVK..DWSKVVLAYEPVWAIGTGKTATPQQAQEVHEKLRGWLKSN
Seq 4  RRAYYGETNEIVADKVAAAVASGFMVIACIGETLQERESGRTAVVVLTQIAAIAKKLKKADWAKVVIAYEPVWAIGTGKVATPQQAQEAHALIRSWVSSK
Seq 5  RRTYHKESDELIAKKFAVLKEQGLTPVLCIGETEAENEAGKTEEVCARQIDAVLKTQGAAAFEGAVIAYEPVWAIGTGKSATPAQAQAVHKFIRDHIA.K

Seq 1  IGTDVAAKLRILYGGSVNAANAATLYAKPDINGFLVGGASLKPEFRDIIDATR.....
Seq 2  LGDKAASELRILYGGSANGSNAVTFKDKADVDGFLVGGASLKPEFVDIINSRN.....
Seq 3  VSDAVAQSTRIIYGGSVTGATCKELASQPDVDGFLVGGASLKPEFVDIINAKQ.....
Seq 4  IGADVAGELRILYGGSVNGKNARTLYQQRDVNGFLVGGASLKPEFVDIIKATQ.....
Seq 5  VDANIAEQVIIQYGGSVNASNAAELFAQHDVDGFLVGGASLKPEFVDIINAAEAAKQA
```

Fig. 4 – Alignment result of *1amk* based on our proposed method.

Table 6 – The objective values of the nondominated solutions for *2hsdA*.

Support	Affine gap penalty	Similarity
18	2064	0.523
	2123	0.546
	2172	0.558
20	2119	0.455
	2188	0.472
	2256*	0.486
22	2256*	0.397
	2284	0.412
	2372	0.449

Table 7 – Test results of four different methods for *2hsdA*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	2372	0.814	0.663
MUSCLE	2381	0.703	0.587
SAGA	2744	0.498	0.362
MSA-GA	2615	0.561	0.408

Table 8 – The objective values of the non-dominated solutions for *1gpb*.

Support	Affine gap penalty	Similarity
3	609	0.484
	617	0.502
	623	0.523
4	649	0.402
	658	0.429
	665	0.455
5	666	0.366
	672	0.383
	683	0.399

Table 9 – Test results of four different methods for *1gpb*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	683	0.992	0.817
MUSCLE	688	0.983	0.802
SAGA	694	0.982	0.802
MSA-GA	716	0.868	0.711

The third experiment investigates the runtime of the proposed method and the other approaches in the case of a data set having large number of sequences. The results plotted in Fig. 5 demonstrate that the performance of MSAGMOGA decreased with respect to the previous data set. This is an expected result since the length of the individuals in the population rises as the sequence number increases in the data set.

The next set of experiments is dedicated to testing the performance of our method on a data set, *1gpb*, having longer sequence lengths as the sequence number is the same with the first data set. Here, as the sequence number of the data set is 5, the sequence length changes in the interval [796,828]. The results are shown in Tables 8 and 9 and Fig. 6. Table 8 gives the non-dominated solutions found by MSAGMOGA for three different values of support. The tradeoff observations are similar to those described above for the two data sets. As can be seen in Tables 8 and 9, in the experiment using larger sequences, MSAGMOGA can find the solution with smaller affine gap penalty than the others. Furthermore, the values of

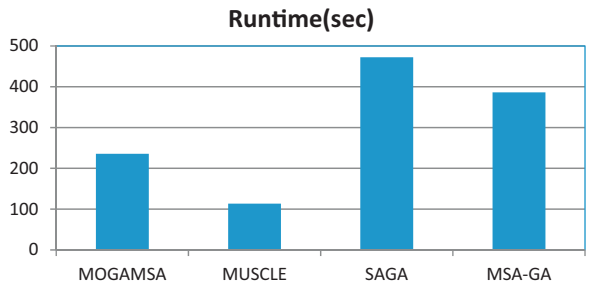

Fig. 5 – Comparison of the runtimes of four different methods for *2hsdA*.

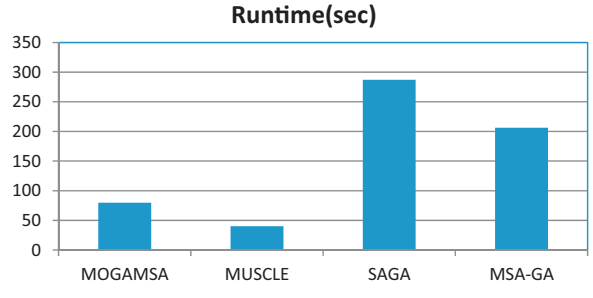

Fig. 6 – Comparison of runtimes of four different methods for *1gpb*.

Table 10 – The objective values of the non-dominated solutions for *Kinase*.

Support	Affine gap penalty	Similarity
3	148	0.574
	162	0.592
	176	0.622
4	184	0.503
	192	0.524
	200	0.569
5	201	0.408
	207	0.442
	219	0.478

Table 11 – Test results of four different methods for *Kinase*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	219	0.881	0.622
MUSCLE	233	0.823	0.578
SAGA	252	0.862	0.601
MSA-GA	414	0.295	0.278

SP and CS of the solution found by MSAGMOGA are not worse than the three other methods.

Finally, Fig. 6 shows the runtime of all the methods. It should be noted that the performance of MSAGMOGA in this data set increased with respect to those of the other data sets and the value of the runtime remained almost the same. This is true because the length of the individuals in the population is proportional to 20% of the length of the longest sequence. But, the runtime of the other methods increases as the length of the sequences increases.

In the fourth set of experiments, we focus on aligning sequences with lower similarity. For this purpose, we used the *Kinase* data set that has almost the same sequence number and length with the first data set, *1amk*, below 25% residue identity. It is important to solve such a problem because recently, most of the efforts to improve the quality of the alignment programs concentrate on the alignment of sequences below 20–25% residue identity. The results shown in Tables 10 and 11 demonstrate that as MSAGMOGA obtains the alternative alignments by a single run, it can also find the quality solution with a lower affine gap penalty whose remaining measures are not worse than the others. The last experiment of this set is dedicated to find the runtimes of all the methods for this new data set with low similarity, *Kinase*. As can be easily seen from Fig. 7, our method exhibited a better

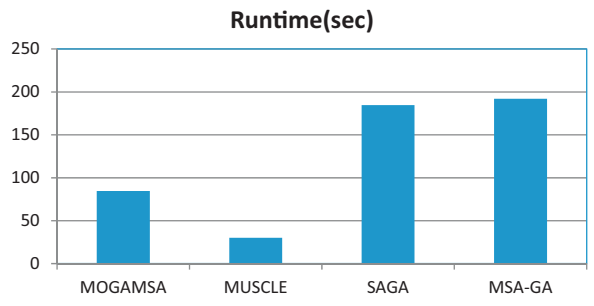

Fig. 7 – Comparison of runtimes of four different methods for *Kinase*.

Table 12 – The objective values of the non-dominated solutions for *1taq*.

Support	Affine gap penalty	Similarity
3	369	0.505
	388	0.524
	409	0.549
4	412	0.437
	426	0.468
	441	0.483
5	443	0.397
	457	0.412
	468	0.428

Table 13 – Test results of four different methods for *1taq*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	468	0.961	0.882
MUSCLE	474	0.946	0.873
SAGA	492	0.931	0.868
MSA-GA	711	0.525	0.431

performance compared to the evolutionary approaches, SAGA and MSA-GA.

In the fifth set of experiments, we considered the *1taq* data set in order to test the weakest case of our method. The features of this data set are reflecting big differences between their sequence lengths, while the sequence number and lengths of this data set are very similar to the *1gpd* data set. The length of the shortest sequence is 806, and the length of the longest sequence is 928. As mentioned in Section 3.1, the length of each individual in MSAGMOGA is proportional to the length of the longest sequence minus the length of the shortest sequence. This leads to the fact that the larger this difference is, the longer the individual is. Table 12 shows the non-dominated solutions found by MSAGMOGA. Here, it can be seen that some solutions were found with very small affine gap penalty when support value = 3. This is true because the longer sequences were ignored for the related experiment. The next experiment deals with comparing the values of the affine gap penalty, SP and CS as it was done earlier. The results are very similar to the previous experiments and confirm that our method found the best solutions as shown in Table 13.

The third and the last experiments of this set investigate the runtime of MSAGMOGA compared to the other approaches for this marginal data set. The results shown in Fig. 8 demonstrate that the result from MSAGMOGA was slightly inferior

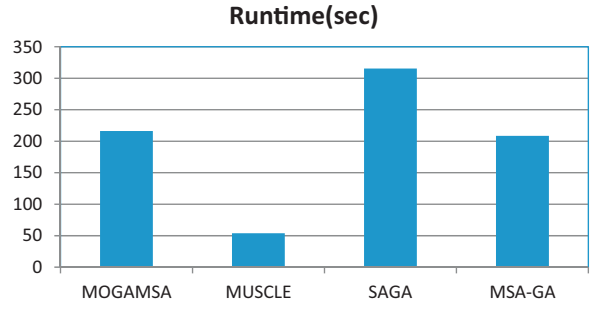

Fig. 8 – Comparison of runtimes of four different methods for *1taq*.

Table 14 – The objective values of the non-dominated solutions for *1dynA*.

Support	Affine gap penalty	Similarity
4	1127	0.318
	1141	0.323
	1158	0.330
5	1176	0.271
	1190	0.283
	1213	0.294
6	1216	0.249
	1229	0.258
	1247	0.266

Table 15 – Test results of four different methods for *1dynA*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	1247	0.042	0.031
MUSCLE	1253	0.033	0.027
SAGA	1276	0.000	0.000
MSA-GA	1354	0.038	0.015

to the MSA-GA's result because of the handicap exhibited by our coding method in the sequences with extremely different length.

In the sixth set of experiments, we handled the 1dynA data set from Ref. [4], having a bigger difference between the shortest and the longest sequence lengths. The results given in Tables 14 and 15 and Fig. 9 demonstrate the effectiveness of our method.

For the last set of experiments, the 2myr dataset has been chosen from Ref. [3]. This dataset includes large number of sequence like 22. Also, the difference between the shortest and the longest sequences is relatively big. We have measured each related criteria of this data set with our method and the other algorithms. The results in Tables 16 and 17 and Fig. 10 are not too different from the previous ones. This case indicates the reliability of MSAGMOGA in obtaining good solutions in even difficult alignment problems (Table 18).

The final experiment yields the statistical significance of the score differences for all aligner pairs. The entries in Table 16 shows *P*-value indicating the statistical significance of the mean scores differences between aligner pairs as measured using Wilcoxon matched-pair signed-rank test. The upper-right corner shows *P*-values calculated using SP scores, and the lower-left corner shows *P*-values calculated using CS scores. *Indicates the aligner on the left gives the worse performance, and the better performance, otherwise. For $P > 0.05$, the difference is considered insignificant and the *P*-value is shown in parentheses. From the table, MSAGMOGA achieves statistically significant accuracy improvement over all the other aligners.

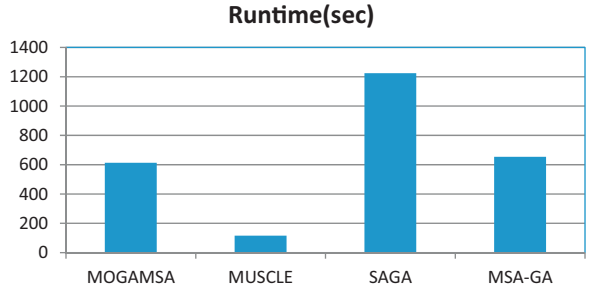

Fig. 9 – Comparison of runtimes of four different methods for *1dynA*.

Table 16 – The objective values of the non-dominated solutions for *2myr*.

Support	Affine gap penalty	Similarity
18	6646	0.324
	6718	0.328
	6802	0.341
20	6851	0.289
	6893	0.296
	6912	0.310
21	6901	0.204
	6936	0.222
	6978	0.244

Table 17 – Test results of four different methods for *2myr*.

Method	Affine gap penalty	SP	CS
MSAGMOGA	6978	0.517	0.483
MUSCLE	7003	0.513	0.478
SAGA	7054	0.321	0.289
MSA-GA	7314	0.215	0.178

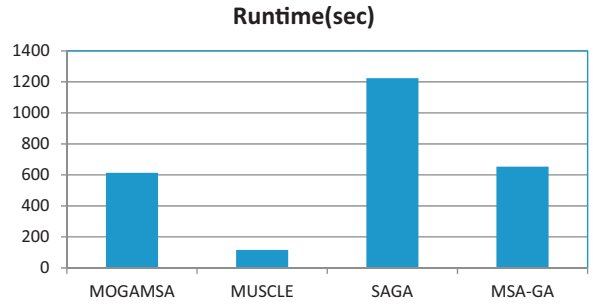

Fig. 10 – Comparison of runtimes of four different methods for *2myr*.

Table 18 – Statistical significance of aligners on BAliBASE 2.0.

	MSAGMOGA	MUSCLE	SAGA	MSA-GA
MSAGMOGA		$<10^{-10}$	$<10^{-10}$	$<10^{-10}$
MUSCLE	$<10^{-10*}$		$<10^{-10}$	$<10^{-10}$
SAGA	$<10^{-10*}$	$<10^{-10}$		0.03*
MSA-GA	$<10^{-10*}$	$2.9 \times 10^{-6*}$	(0.13)	

5. Discussion and conclusions

In this paper, we contributed to the ongoing research by proposing a multi-objective GA based method, MSAGMOGA, for finding optimized alignments with affine gap according to the criteria we defined. These criteria are similarity of sequences, affine gap penalty and support exhibiting the strength of alignment.

MSAGMOGA includes five contributions. First, the algorithm is equally applicable to any type of sequential data. Second, it allows arbitrary similarity measure. Although we used a relatively simple similarity measure in the paper, it can be easily changed or extended. Another contribution is that a large number of nondominated sets are obtained by its single run. Thus, the decision maker can understand the tradeoff between the similarity, affine gap penalty and support by the obtained alignment. Next, by MSAGMOGA, we proposed an efficient coding method for multi-objective genetic algorithm. Fourth, the optimal alignments are obtained by introducing the support term for the multiple sequence alignment problem. Finally, MSAGMOGA outperforms all the methods compared in this paper in terms of accuracy and statistical significance, and the two well-known evolutionary methods, SAGA and MSA-GA, in terms of runtime. The method spends less time in especially longer sequences when it is compared with the other methods.

The experiments conducted on five data sets illustrate that the proposed approach produces meaningful results and has reasonable efficiency. The results for the five data sets are consistent and hence encouraging. MSAGMOGA can be also directly applied to other diverse types of sequential data sets, or it can be extended to address problems not yet considered.

In the future, we will revise our similarity measure to make this score more realistic, improve our algorithm such that one could have better performance in lower similar sequences, and experiment with realistic data sets having longer sequences and large number of sequences.

Conflict of interest

The authors do not have conflict of interest to declare.

REFERENCES

[1] C. Notredame, Recent progresses in multiple sequence alignment: a survey, Pharmacogenomics 3 (1) (2002) 131–144.

[2] L. Wang, T. Jiang, On the complexity of multiple sequence alignment, Journal of Computational Biology 1 (4) (1994) 337–348.

[3] P. Bonizzoni, G.D. Vedova, The complexity of multiple sequence alignment with SP-score that is a metric, Theoretical Computer Science 259 (1–2) (2001).

[4] S.K. Gupta, J.D. Kececioglu, A.A. Schaffer, Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment, Journal of Computational Biology 2 (3) (1995) 459–472.

[5] P. Hogeweg, B. Hesper, The alignment of sets of sequences and the construction of phyletic trees: an integrated method, Journal of Molecular Evolution 20 (1984) 175–186.

[6] G.J. Barton, M.J.E. Sternberg, A strategy for the rapid multiple alignment of protein sequences, Journal of Molecular Biology 198 (1987) 327–337.

[7] J.D. Thompson, T.J. Gibson, F. Plewniak, F. Jeanmougin, D.G. Higgins, The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools, Nucleic Acids Research 24 (1997) 4876–4882.

[8] J.D. Thompson, D.G. Higgins, T.J. Gibson, W. Clustal, improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, Nucleic Acids Research 22 (1994) 4673–4680.

[9] C. Notredame, D.G. Higgins, J. Heringa, T-Coffee: a novel method for fast and accurate multiple sequence alignment, Journal of Molecular Biology 302 (2000) 205–217.

[10] R.C. Edgar, MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Research 32 (5) (2004) 1792–1797.

[11] K. Katoh, K. Kuma, H. Toh, T. Miyata, MAFFT version 5: improvement in accuracy of multiple sequence alignment, Nucleic Acids Research 33 (2005) 511–518.

[12] C.B. Do, M.S. Mahabhashyam, M. Brudno, S. Batzoglou, ProbCons: probabilistic consistency-based multiple sequence alignment, Genome Research 15 (2) (2005) 330–340.

[13] H. Hongwei, V. Stojkovic, A simulated annealing algorithm for multiple sequence alignment with guaranteed accuracy, in: Third International Conference on Natural Computation. ICNC 2007, 2007.

[14] F.J.M. Silva, J.M. Sánchez-Pérez, J. Antonio, G. Pulido, M.A. Vega-Rodríguez, An evolutionary approach for performing multiple sequence alignment, in: IEEE Congress on Evolutionary Computation (CEC), 2010.

[15] D.W. Mount, Using hidden Markov model to align multiple sequences, in: Bioinformatics: Sequence and Genome Analysis, 2nd ed., CSHL Press, Cold Spring Harbor NY USA, 2004.

[16] Lawrence, et al., Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment, Science 262 (5131) (1993) 208–214.

[17] T. Lassmann, E.L.L. Sonnhammer, Quality assessment of multiple alignment programs, FEBS Letters 529 (2002) 126–130.

[18] C. Notredame, D.G. Higgins, SAGA: sequence alignment by genetic algorithm, Nucleic Acids Research 24 (1996) 1515–1524.

[19] C. Gondro, B.P. Kinghorn, A simple genetic algorithm for multiple sequence alignment, Genetics and Molecular Research 6 (4) (2007) 964–982.

[20] J. Taheri, A.Y. Zomaya, RBT-GA: a novel metaheuristic for solving the multiple sequence alignment problem, BMC Genomics 10 (2009) 1–11.

[21] A. Bahr, J.D. Thompson, J.-C. Thierry, O. Poch, BALIBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutation, Nucleic Acids Research 29 (1) (2000) 323–326.

[22] F. Naznin, R. Sarker, D. Essam, Progressive alignment method using genetic algorithm for multiple sequence alignment, IEEE Transactions on Evolutionary Computation 16 (October (5)) (2012) 615–631.

[23] P. Seeluangsawat, P. Chongstitvatana, A multiple objective evolutionary algorithm for multiple sequence alignment, in: GECCO'05, 2005.

[24] A. Taneda, Multi-objective pairwise RNA sequence alignment, Bioinformatics 26 (19) (2010) 2383–2390.

[25] F.J. da Silva, J.M. Pérez, J.A. Pulido, M.A. Rodríguez, Parallel Niche Pareto AlineaGA—an evolutionary multiobjective approach on multiple sequence alignment, Journal of Integrative Bioinformatics 8 (3) (2011) 174.

[26] F. Ortuno, J.P. Florido, J.M. Urquiza, H. Pomares, A. Prieto, I. Rojas, Optimization of multiple sequence alignment methodologies using a multiobjective evolutionary algorithm based on NSGA-II, in: IEEE Congress on Evolutionary Computation (CEC), 2012.

[27] K. DeRonne, G. Karypis, Pareto optimal pairwise sequence alignment, IEEE/ACM Transactions on Computational Biology and Bioinformatics 10 (March (2)) (2013) 481–493.

[28] E. Zitzler, et al., Comparison of multi-objective evolutionary algorithms: empirical results, Evolutionary Computation 2 (2000) 173–195.
[29] M. Kaya, M. Güç, A novel approach to extract structured motifs by multi-objective genetic algorithm, in: The 21th IEEE International Symposium on Computer-Based Medical Systems, Jyväskylä, Finland, June 17–19, 2008.
[30] M. Kaya, Motif discovery using multi-objective genetic algorithm in biosequences, in: 7th International Symposium on Intelligent Data Analysis (IDA), Ljubljana, 6–8 September, 2007.
[31] K. Deb, et al., A fast and elitist multi-objective genetic algorithm: NSGA II, IEEE Transactions on Evolutionary Computation 6 (2002) 182–197.
[32] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[33] K. Chellapilla, G.B. Fogel, Multiple sequence alignment using evolutionary programming, in: Proceedings of the First Congress of Evolutionary Computation, 1999, pp. 445–452.
[34] M.O. Dayhoff, Survey of new data and computer methods of analysis, in: M.O. Dayhoff (Ed.), Atlas of Protein Sequence and Structure 5, Suppl. 3, National Biomedical Research Foundation, Silver Springs, 1978, pp. 2–8.
[35] S. Henikoff, J.G. Henikoff, Amino acid substitution matrices from protein blocks, Proceedings of the National Academy of Sciences 89 (1992) 10915–10919.