International Journal of Computational Intelligence and Applications
Vol. 11, No. 4 (2012) 1250023 (18 pages)
© Imperial College Press
DOI: 10.1142/S146902681250023X

Imperial College Press
www.icpress.co.uk

A NEW GENETIC ALGORITHM FOR MULTIPLE SEQUENCE ALIGNMENT

ZAHRA NARIMANI*, HAMID BEIGY† and HASSAN ABOLHASSANI‡

Department of Computer Engineering
Sharif University of Technology, Tehran, Iran
*narimani@ce.sharif.edu
†beigy@sharif.edu
‡abolhassani@sharif.edu

Received 14 May 2011
Revised 23 January 2012
Published 14 December 2012

Multiple sequence alignment (MSA) is one of the basic and important problems in molecular biology. MSA can be used for different purposes including finding the conserved motifs and structurally important regions in protein sequences and determine evolutionary distance between sequences. Aligning several sequences cannot be done in polynomial time and therefore heuristic methods such as genetic algorithms can be used to find approximate solutions of MSA problems. Several algorithms based on genetic algorithms have been developed for this problem in recent years. Most of these algorithms use very complicated, problem specific and time consuming mutation operators. In this paper, we propose a new algorithm that uses a new way of population initialization and simple mutation and recombination operators. The strength of the proposed GA is using simple mutation operators and also a special recombination operator that does not have problems of similar recombination operators in other GAs. The experimental results show that the proposed algorithm is capable of finding good MSAs in contrast to existing methods, while it uses simple operators with low computational complexity.

Keywords: Multiple sequence alignment; genetic algorithms; fitness function; genetic operators.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

1. Introduction

Analyzing biological sequences is one of the important areas of study in biology. One of the basic problems in this area is alignment of protein and DNA sequences. This problem is called sequence alignment and when more than two sequences are to be aligned, the problem is called multiple sequence alignment (MSA). There is no polynomial time algorithm for solving MSA and several approximate methods have been used to find near optimal solutions. MSA can be defined formally as the following.[1] Given n sequences $S = \{s1, s2, \ldots, sn\}$, defined on alphabet A where each sequences i has length of li. Let a new alphabet $A' = A \cup \{-\}$, where "–" is a gap

character. A new sequence set S′ defined on A′ is called an alignment if the following conditions hold:

(i) All sequences in S′ have the same length, l′, such that $\max_{1\leq i\leq n} l_i \leq l' \leq \sum_j l_j$.
(ii) Omitting gap characters, each sequence in S′ is equal to its equivalent sequence in S.
(iii) If we align sequences in S′, there is no column containing all gap characters.

So we can consider a multiple sequence alignment as a matrix of n rows, each row containing one sequence with extra gap characters. In the case of DNA sequences, the alphabet A consist of four nucleotides (A, T, C, G) and in the case of protein sequences the alphabet consist of twenty amino acids. If there are only two sequences, the alignment is called a pairwise alignment, and if number of sequences is more than two, the alignment is a multiple sequence alignment. Figure 1 shows some examples of pairwise and multiple sequence alignment on DNA sequences.

The quality of an alignment of protein sequences is measured by alignment score, which is a function of aligned amino acids scores. Amino acid alignment scores can be gained from substitution matrices like PAM or BLOSUM. Usually a gap penalty is considered for amino acids that aligned with gaps in other sequences. Alignment score may be defined as sum of scores of amino acid pairs aligned together, and a gap penalty is usually decreased from this score. Some examples of alignment scoring can be seen in Refs. 2, 3 and 4. As mentioned before, MSA cannot be solved in polynomial time and using exact methods like dynamic programming for solving MSAs is not computationally feasible. Therefore other methods are used for finding approximate solutions for this problem. Progressive algorithms, evolutionary algorithms and hidden Markov models (HMM) methods are some methods to find approximate MSAs.

In progressive methods, dynamic programming is used to incrementally align sequences. Once a sequence is added to the alignment, gaps cannot be removed from it anymore. So if there are misalignments in early steps, they are remained in final alignment too. Due to this greedy approach that may create a wrong solution which

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

1	C A - T G A G - A T
2	- A C T C A G T A -

(a)

1	C A - T G A G - A T
2	- A C T C A G T A - C
3	C A - A G - G - A T C
4	G - - T C A G T A - C

(b)

Fig. 1. Example of (a) pairwise alignment on DNA sequences s1 = CATGAGATC and s2 = ACTCAGTAC (b) and multiple alignment on DNA sequences s1 = CATGAGATC, s2 = ACTCAGTAC, s3 = CAAGGATC and s4 = GTCAGTAC.

cannot be corrected in further steps of the algorithm, the order in which sequences are added to the alignment, is very important. The popularity of progressive methods is due to their good running time. This is why sometimes alignments gained from a progressive algorithm are used as a seed in population based MSA methods. MULALIGN,[5] MULTUL,[6] PILEUP,[7] ClustalW,[8] Muscle[9] and T-Coffee[10] are some examples of progressive algorithms for multiple sequence alignment. Some progressive methods such as ClustalW, MULTALIGN and PILEUP use a guide tree for making alignments. The former method uses neighbor-joining method[11] for making the tree, and the two later algorithms use UPGMA.[12] The guide tree can help progressive alignment to avoid doing pairwise alignment on all of existing sequence pairs.

Evolutionary methods are one of iterative methods used to solve MSAs. Evolutionary methods use an initial population of potential solutions and try to improve the population (and maximum fitness) using mutation and crossover operators. Finally, after reaching stopping criteria (like maximum fitness, or maximum number of generations) the best member of the population is introduced as the answer of problem. Evolutionary methods are very flexible because they have no special limitation on objective function to be optimized; they can correct misalignments of early generations and they can use primary knowledge (as seeds) in the initial population. SAGA,[4] IterAligne,[13] MSA-EA,[3] AlineaGA[14,15] are examples of evolutionary algorithms for MSA. The problem with existing evolutionary algorithm is their use of complicated operators which make the algorithm computationally expensive. Methods for generating initial population and also for crossing over and mutating the population members in these algorithms are described in Sec. 2.

HMM is also one of iterative methods for solving MSAs. MUMMALS[16] and ProbAlign[17] are examples of HMM solutions for MSA. HMM is a probabilistic model which is used to align a new sequence with an existing alignment. The sequences are added one after another to the model to form the multiple sequence alignment. This is useful for example in cases where we want to check if a protein sequence belongs to a protein family or not. The main drawback of HMM is that the number of sequences should be large enough in order to create a statistical model.

In this paper, a new genetic algorithm is proposed for solving MSA problem that uses simple mutation and recombination operators. The strength of the proposed GA is because of using simple mutation operators and also a special recombination operator that does not have problems of similar recombination operators in other GAs. The new genetic algorithm is implemented in java and the results are tested on BAliBASE2.01dataset.[18] Results of experiments show that this algorithm is comparable to other algorithms in BAliBASE column score and sum-of-pairs score, while its running time is considerably less than other genetic algorithms for finding MSAs. The considerable point in the result is that it shows it is not necessary to use complicated operators for solving complicated problems like MSA, but it is necessary to define precise operators that are able to improve genetic population and transfer good building blocks to the next generations.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

The rest of this paper is organized as follows: In Sec. 2 we have a review on related works and our motivation in the proposed method. In Sec. 3 the proposed method is explained and experimental results are presented in Sec. 4. In Sec. 4, two other methods are compared with our algorithm, the first experiment compares the new method with two other genetic algorithmsMSA-GA[2] and the algorithmVDGA_Decomp proposed in Ref. 15 and progressive algorithmClustalW[8] on BALIBASE2.0 dataset, and the second experiments compares the number of matched columns in final alignments in the new method withAlineaGA[14] and T-COFFEE.[10] AlineaGA algorithm concentrates on finding fully matched columns and in the result we compare the number of fully matched columns with it. Section 5 concludes the paper.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

2. Related Works

In this section, we briefly review existing genetic algorithms for solving MSA problem. The method of population initialization and defined mutation and recombination operators are important elements of a genetic algorithms. In this section, these elements are reviewed in existing genetic algorithms and their effect on the performance of the algorithm and quality of result is discussed.

2.1. *Mutation and crossover operations*

Several different genetic algorithms have been developed for solving MSA in the literature. Because of the complicated nature of the problem, most of these methods use complicated operators, and of course a complicated algorithm with high running time. SAGA,[4] which is one of the firs effective ones, used different complicated operators with special operator scheduling. In other algorithms like GA in Ref. 19, MSA-EA,[3] PRALINE,[20] and AlineaGA method[14,21] the same technique is used; operators are very complicated and local search methods are used to bias the operations toward finding a potentially good alignment. Some of these operators try to find local similarities and align these areas in different sequences and some try to find fully matched columns in alignment. In Ref. 15, single point and multiple point crossover were used. These crossover operators select one or several random points as a vertical crossover point in two parents. In single point crossover the first part of the child is selected from the first vertical half of the parent having bigger local score in that half and the sequences are completed from the other part of the second parent, in order to complete the first part. In multiple point crossover with two cut point for example, the first and last parts of the child is selected from the parent which have higher score in that parts, and the part between these two part, is selected from the other parent in order to complete the sequences. The method introduced in Ref. 15 uses guide trees in its mutation operator. For the selected member for mutation, which is a MSA, a distance table[22,23] is calculated. Then a new guide tree is built on these sequences. Sequence numbers are shuffled in the guide tree generation each time in order to gain a better guide tree. The process of guide tree generation is

repeated until a successful tree is generated or the maximum threshold of 50 unsuccessful attempts has done. In this method, authors have used vertical decomposition for breaking the alignments into sub alignments, and then merge these sub alignments to gain the full alignment again. This helps in aligning long sequences as we will see in Sec. 4.1 in experimental results.

Beside these complicated algorithms, we found the method MSA-GA[2] used simple and random operators but produced acceptable final results. This was our motivation to study this method and see why it has great result and why other algorithms have to use complicated operators if simple ones also will work properly in this problem. The main difference is its objective function. Normally, the objective function used for MSA is a weighted sum of pair of amino acid alignment scores minus a gap penalty. Gap penalty is necessary to avoid inserting lots of gaps in the sequences and it makes the objective function a better approximate of biological alignment fitness.

Considering gap penalty in the objective function, when a gap is inserted into an alignment, its score is decreased. So mutated genes with gap insertion will have a lower fitness than what they had before mutation. Because of this reason, mutation operators in GAs considering gap penalty, have to be somehow biased to compensate this fitness reduction. If these operators are not designed specifically for finding better aligned columns, mutated genes will have a decreased fitness and therefore less chance for being transferred to future generations. This will actually repress mutation operator and interfere its affectivity. In fact the ability of technique given in MSA-GA to improve generations without using complicated operators is because they omit gap penalty in the objective function. Although omitting gap penalty leads to a worst objective function from biological point of view, it gives the freedom to mutation operator to put its effect in the genes and let the mutated genes to have an improved fitness. The experimental results given in Fig. 2 show that if gap penalty is added to this objective function, the algorithm will not act as good as previous and even in cases that it uses an initial CLUASTALW seed it will remain in the local

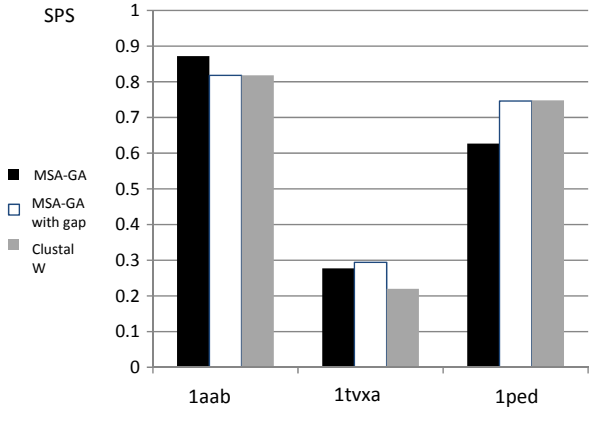


Fig. 2. Gap penalty effect on sum of pairs score (SPS) in Gondro and Kinghorn's method (MSA-GA).

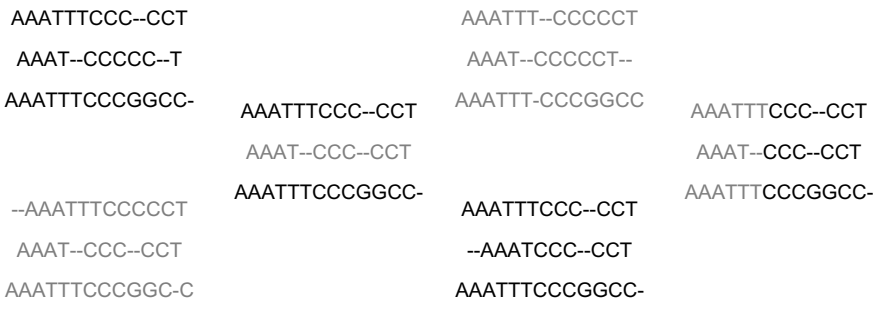
Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

maximum around seed score. This is because of this fact that seed has a high fitness value and inserting gaps in it leads to a decrease in fitness. So genes gained from a mutated seed would not be chosen as the best gene of the population and the maximum found by GA remains at seed. These results are shown on sequence sets 1aab, 1tvxa and 1ped from BAliBASE. In Fig. 2, we can see that two sets 1aab and 1ped are remained at local optima of seed after adding gap penalty to the fitness function. This is because there is not a complicated mutation or crossover operation that can improve the seed score enough to overcome gap penalty decrease in the objective function, and even if gaps are removed from the seed by these operators they lead to a worst score because seed is an approximate alignment with a good fitness that omitting gaps randomly from it more likely decreases its fitness. In 1ped, the result is even worse than seed (although the fitness function has increased) since omitting gap from fitness function makes it a worst approximate of biological facts than a fitness function using gaps. In MSA-GA, the authors have used different parameter setting for this sequence set in order to improve the results. In 1tvxa both fitness functions are not good approximates for a biological fitness value, so the results are not good. But we can see that adding gap penalty to fitness function have a positive impact on the result in this sequence set. In other sets we have the similar observation; in cases that fitness function without gap penalty resulted in better alignments, adding gap penalty to it made the final results a little worse than non-using gap state.

But what is the problem with complicated operators? The first problem is that these operators are so time consuming. Furthermore, they are biased toward generating a special property in the population members (like trying to generate fully matched columns in the new) and there is no fact that guarantee the final answer is also good when having fully matched columns. In fact in a typical GA, operators are applied on population to improve individuals and to transfer building blocks to future generations. A crossover operation is successful if it can generate an offspring with more building blocks than its parents.[24] However, in these complicated operators, genes are manipulated in a special direction and building blocks are changed after applying the operators. So these operators are not necessarily good genetic operators. Some algorithms like AlineaGA use complicated biased recombination operators too. It is shown in Ref. 25 that SAGA is highly dependent on its complicated mutation operators and ordinary crossover has not a special role in this method. In fact these are biased operators that play important role in these algorithms. In MSA-GA, two crossover operators are defined; horizontal recombination [Fig. 3(a)] and vertical recombination [Fig. 3(b)]. It selects each sequence in offspring randomly from one of the parents [Fig. 3(a)]. The vertical recombination, which needs more attention, chooses two genes (alignments), and randomly specifies a cut point for recombining the parents. The offspring is made by merging the parents, one from index 1 to cut point and the other one from cut point to the end of the sequences. The recombination operator merges the parents if their two parts are compatible.

Vertical recombination seems to have no problem at first. It ensures integrity of sequences because of its compatibility check. But this operator merges two genes just

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.



(a) Horizontal recombination (b) Vertical recombination

Fig. 3. Horizontal and vertical recombination defined in MSA-GA.[2]

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

if they are compatible; this shows that in the marginal cut points (near to start or end of sequences) there is more possibility to have two compatible genes than innermost points because the number of amino acids to be equal is less in these marginal cut points. In fact selected cut points for recombination are happening more likely at marginal indexes. As Fig. 4 shows, the average number of vertical recombination that has occurred in each index — on sequences from Ref. 1, 1aab set of BAliBASE — approves our statement. In this diagram, when a cut point is not suitable, the selection is repeated until an appropriate index is selected. In other subsets of BAliBASE a similar result was also obtained.

This is a serious problem that biases crossover operation. In many cases, the crossover will choose the start point of genes to merge them, i.e., no recombination is done. In contrast to this vertical recombination, another vertical recombination proposed in SAGA.[4] This operator selects a cut point and chooses the first part of offspring from one of the parents. For the other part, sequences from the second parent (which is not used for the first part of offspring) are selected in a way that can be joined to right side of the first part's sequences — it means that sequences may be selected from different starting indices in the second part. This operator does not have the problem of biased cut points but it has a more important problem of disruption of building blocks. The problem is that because sequences from the second parent are selected from different indices and the aligned sequences are not inherited

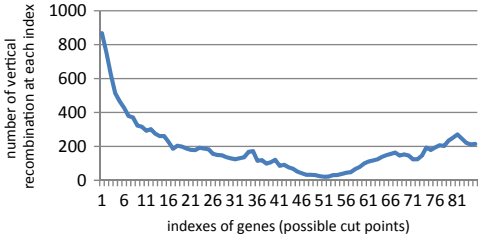


Fig. 4. Average number of successful vertical recombination at each index of alignments in MSA-GA.

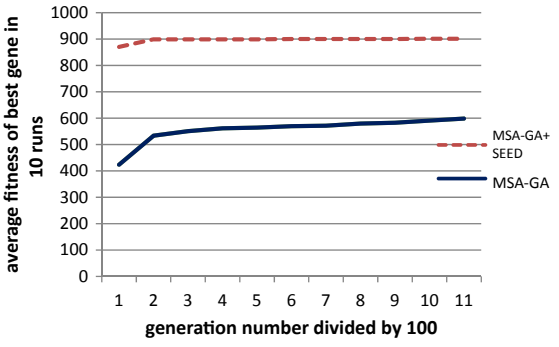


Fig. 5. Average fitness of the best gene MSA-GA with/without seed.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

to offspring anymore. The problem of building blocks disruption is called linkage problem in Ref. 26. In uniform crossover operator, which is defined in SAGA, disruptive nature of this single point crossover also exists. This operator is designed to swap blocks which are between two consistent positions. From these studies we conclude that complicated and biased operators are not necessarily good for GA, especially in MSA case which the fitness function is itself an approximation of an unknown fact of evolution.

2.2. *Population initialization*

Some GAs for MSA, use random population initialization. This initialization method leads to late convergence of GA to a good solution, because the search space for this problem is very large (all possible alignments). In a few methods, an initial seed, gained from a progressive approach, is added to the initial population and therefore the algorithm task is to improve the seed[2,3]; however using an initial seed can cause the algorithm to entrap in local optima.[3,25]

In MSA-GA,[2] a special method is used for population initialization. Instead of creating random sequences, each sequence in a gene is selected from the set of all its pair-wise alignments with other sequences. This method has a good effect on the final result, but as we can see in Fig. 5 that in this algorithm final answer is highly dependent on its initial population. In fact it is the initial population that leads to a good result. It can be seen that the specific method for population initialization lead to a good score at the first populations and operators cannot improve this population much further; this can be the problem of entrapping in local optima mentioned in Refs. 3 and 25. As a conclusion we can say that although randomness is not good for initial population in a large problem space like what we have in MSA, using a very specific initial population may also not be a good solution.

3. The Proposed Method

In this section, we propose a new GA based algorithm, which is given in Fig. 6, for solving MSA problem. The proposed idea is to define a new way for population

1) Generate the first generation of n MSAs
2) Evaluate alignment score for each chromosome
3) While true
   (a) Create a new generation by applying new operators and selection method; each gene in the new generation is selected by applying cross over or mutation operators on parents. Operators are chosen according to probability parameters for each.
   (b) If termination criteria(maximum number of generations) is reached break

Fig. 6. The proposed genetic algorithm for MSA.

initialization and also a new operator that does not have problems of existing operators. Our goal in designing new operators is to have simple, unbiased operators, which are able to transfer building blocks to further generations. By unbiased we mean that we do not want our operators to specially work on a specific goal, like creating fully matched columns; because in addition to adding to complexity, these biased operators usually destroy building blocks, also this approach of using biased operators is a greedy local method which may not necessarily improve the whole alignment. In this section, we first explain our genetic algorithm and then we specifically explain our new method for population initialization and operator definition in the following subsections.

Gene representation: Each individual in the population is represented as a matrix. In some of existing methods a maximum length is set for alignments. This will manage sequence lengths and avoids alignments to be unlimitedly long. We use the same maximum length for alignments which is 20% longer than the longest sequence length.[2]

Population initialization: Each gene in the population is a possible alignment. These alignments are generated in a new way of population initialization which will be described later in this section.

Mutation and recombination operators: Mutation operators used in algorithm consists of: openGap (which opens a new gap in a randomly selected sequence of the gene), extendGap (which chooses an arbitrary gap block from a gene and extends it by adding one gap character), and reduceGap (which select an existing gap block from the gene and reduces its size). The only crossover operation we used in our algorithm is a new operator which is designed to be simple and also useful for multiple sequence alignment problems. Description of this operator is given later in this section.

Objective function: The objective function is sum-of-pairs score (SPS). Also a gap penalty is decreased from this score in order to prevent from too much gap insertion in genes.

Gene selection: We keep a predetermined percent of top genes in each selection phase. This percent is a parameter (the parameter setting for our result is specified in

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

result section). Other genes are selected by performing crossover with the determined probability. If no crossover were done, one of the genes will be mutated with a specific probability and copied to the next population. Finally, this gene will be copied to the next generation if no mutation is performed too.

Any population initialization method, mutation or crossover operators, and selection method may be used in this algorithm generally. In Step 1, first generation is initialized (In Sec. 3.1 our method for population initialization will be discussed). In each gene selection phase, we use the gene selection method for creating the next generation. If a crossover has to be applied on a gene in gene selection, a second phase of choosing crossover method type is done. In this phase, a random number is generated; if it is less that horizontal recombination probability, this recombination is applied on genes and the crossover method returns the child with larger fitness, else another number is generated and if it is less than vertical recombination probability this recombination is applied and a child will be returned. Horizontal and vertical recombinations will be described later in Sec. 3.2. The same process is done to select the appropriate mutation operator; we use openGap, reduceGap and extendGap probabilities instead. Finally, the genetic algorithm terminates after a certain number of generations.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

3.1. *Population initialization*

As mentioned before, using an initial population that is not very limited and neither completely random is appropriate for problem of MSA. Also we saw that using an initial population with a seed, could increase the probability of entrapping GA in local optima and specialize initialization of genes will restrict the search space. Here, we give a new method, as given in Fig. 7, for population initialization that is anintermediate method which is not completely random, and also not very determined.

In this figure, L is the length of the largest sequence and M is the size of the smallest sequence, and seedProb is a predetermined parameter showing the probability of using seed. Seed is an alignment of sequences by ClustalW method. The number of gaps is determined experimentally. Initially seedProb in most of our experiments is selected such that in every two gene of the population, there exists at least one sequence of initial seed. For small number of sequences, it, was necessary to

For each gene in the initial population:
For each sequence in the current gene:
- Choose a random number with uniform distribution in the interval [0, 1]
- If random number is less than the probability parameter, seedProb, select this sequence from seed and put it into current gene
- Else generate a random number of gaps from interval [0, (L-M)/2] and add it to the original sequence; then put this new sequence in the gene

Fig. 7. The proposed algorithm for population initialization.

decrease this value, because it caused the population to be uniform. But in larger sequence sets, like 1taq, even choosing value of 0.5 (a large value) had positive effects on results.

3.2. *Operators*

In the proposed method, we want to use just simple operators. The mutation operators are defined as: openGap (which opens a new gap in a randomly selected sequence of the gene), extendGap (which chooses an arbitrary gap block from a gene and extends it by adding one gap character), and reduceGap (which select an existing gap block from the gene and reduces its size). In designing the crossover operator we consider two objectives. First we should be careful about linkage problem (discussed in Sec. 2.1); we do not want the proposed crossover to disrupt building blocks. Second we want the operator to be less computationally expensive.

We propose a new crossover operator that does not have biased indexes problem (discussed in Sec. 2.1) and we have tried to minimize linkage problem in it. The new operator is called cross-mutate because it is a combination of crossover and mutation operators; and operates as given below:

- For parents P1 and P2, choose a cut point randomly
- Copy the first part of offspring from P1, and the second part from P2 to generate child C1, and copy the first part of P2 and the second part of P1 to generate child C2.
- Mutate the redundant amino acids (which are repeated in the second part too) in the first parts and convert them into gap characters.
- If some amino acids are omitted from offspring, choose equal number of gap characters from the first part respectively from cut point to start; and put those amino acids instead of gap characters respectively.
- Select the child with highest score as the final offspring.

The proposed cross-mutate operator ensures integrity of sequences, because the number and order of amino acids are not changed in the offspring, and it does not have biased cut point problem because it performs the recombination at every selected index. It slightly changes the alignment in the first part but does not change the whole structure of the alignment gained by the first parent, therefore its linkage problem is very much less than that of single point crossover in SAGA. An example of cross-mutation is shown in Fig. 8. In this figure, C2 is the child with larger score and therefore is selected as the final offspring.

Wherever a column consisting of only gap characters is visited in genes, it is removed by a pruning operation. The pruning operator also checks whether all sequences in the alignment have the same length, and adds gap characters to make them equal in length.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

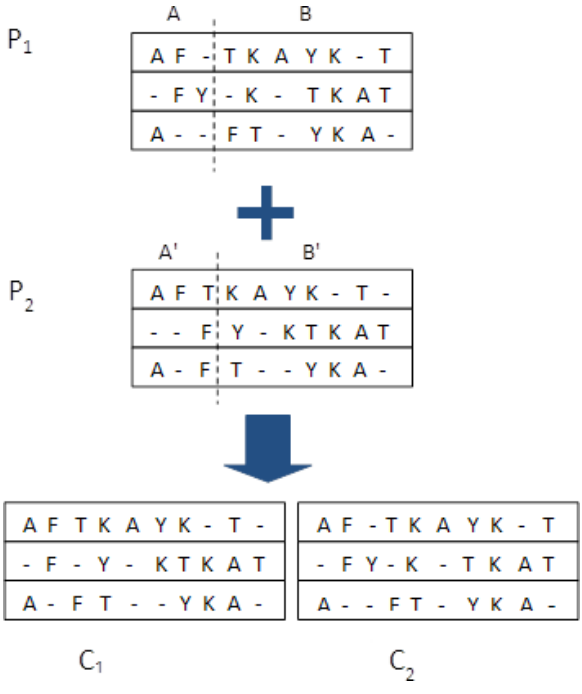


Fig. 8. New cross-mutate operator.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.

4. Experimental Results

In order to evaluate performance of the proposed algorithm, we have performed some experiments and compared the proposed algorithm with some related algorithms. The proposed algorithm is implemented in Java and the results are tested using BAliBASE2.01 dataset.[18] This free dataset is a collection of semi-automated and human made alignments and therefore is not biased toward any specific method. We use its sequences in our experiments and final results are compared to alignments in this reference set. SPS (sum-of-pairs score) from baliscore program available in BAliBASE is used to compute alignment scores for comparing final results.

We have two sets of experiments. In the first one, we compare SPS scores with some existing algorithms. This is explained in Sec. 4.1, in which current method is compared to genetic algorithms MSA-GA and VDGA_Decomp,[15] and also ClustalW. This comparison is done on BALIBASE2.0 data and based on SPS values of final alignments. In the second experiment, Sec. 4.2, results are compared with AlineaGA and T-COFFEE. The criteria for comparison in this experiment is the number of matched columns in final alignments. Fitness values are not compared in this section because we were not sure about gap opening and gap extension penalty. Sequences in this section are subsets 1, 2 and 3 (Subset 1: Human alpha and beta Hemoglobin; Subset 2: Subset 1 sequences, Duckbill platypus alpha and beta Hemoglobin;

Subset 3: Subset 2 sequences, Anasplatyrhynchos alpha and beta hemoglobin) selected from Uniprot database[27] in Ref. 14 for testing results. We are using this comparison to compare the number of matched columns in the proposed method which uses very simple operators, with AlineaGA which its operators and even objective function are completely designed toward finding matched columns.

4.1. *Comparison of the results with MSA-GA and VDGA_Decomp*

Sequences selected in this experiment are a subset of selected sequences in Ref. 2. Reference sets are selected in three main categories according to the sequence identity;

- Less than 25% identity: 1tvxa (small), kinase (medium), 1ped (large)
- 20–40% identity: 1ses (small), 1ad2 (medium), 1ycc (large)
- More than 35% identity: 1krn (small), 1amk (medium), 1taq (large)

In Table 1, average and best result of proposed method is given on selected data. This result is demonstrated using SPS (Sum of Pairs Scores: the portion of pairs which are aligned according to reference alignment) and CS (Columns score: the portion of columns in final alignment which is completely matched to the reference alignment) score. Average results are gained from 20 run with omitting outliers. Best scores are given by selecting the best score in 10 independent runs. The new method uses 2000 genetic generations in each run.

In Table 2 best SPS results of new method are compared to MSA-GA and VDGA_Decomp. Results from MSA-GA and VDGA_Decomp are chosen from experimental results given in Ref. 15. MSA-GA result is demonstrated on two columns, one without ClustalW seed and the other with pre aligned seed. In VDGA_Decomp 2, 3 and 4, vertical decomposition decomposes the alignments into 2, 3 and 4 parts, respectively. VDGA_Decomp3 is the best setting of VDGA in the results of the original paper and if we omit its result our algorithm would have the best result in most of sets.

Table 1. Average and best SPS and CS of proposed method on BAliBASE2.01 dataset (number of generations = 2000).

	Avg SPS	Best SPS	Avg CS	Best CS
1aab	0.856	0.872	0.781	0.804
Tvxa	0.267	0.301	0	0
Kinase	0.640	0.662	0.484	0.519
1ped	0.720	0.785	0.650	0.721
1ycc	0.662	0.712	0.429	0.560
1ad2	0.888	0.904	0.792	0.804
1sesA	0.913	0.941	0.810	0.871
1krn	0.947	0.982	0.883	0.954
1amk	0.960	0.975	0.910	0.937
1taq	0.764	0.823	0.525	0.724

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

Table 2. Comparing maximum fitness gene SPS values on BAliBASE2.01 dataset (number of generations = 2000) with best results of MSA-GA and VDGA_Decomp.

	MSA-GA	MSA-GA W PreAlign	CIastalW	VDGA_ Decomp2	VDGA_ Decomp3	VGDA_ Decomp4	Proposed Method
tvxa	0.295	0.209	0.042	0.316	0.316	0.310	0.301
Kinase	0.295	0.488	0.479	0.531	0.545	0.548	0.662
1ped	0.501	0.687	0.592	0.443	0.482	0.451	0.785
1ycc	0.650	0.653	0.643	0.752	0.839	0.685	0.712
1ad2	0.821	0.843	0.773	0.939	0.950	0.941	0.904
1sesA	0.620	0.913	0.913	0.917	0.962	0.923	0.941
1krn	0.908	0.825	0.893	0.942	0.960	0.892	0.982
1amk	0.963	0.959	0.943	0.982	0.984	0.982	0.975
1taq	0.525	0.826	0.826	0.938	0.959	0.944	0.823

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

We can see that in kinase, 1ped and 1krn our algorithm shows highly improved results over other methods. In sets 1amk and 1sesA our method does not perform better, but near the best method. Still in these cases our method performs better that MSA-GA. In 1taq our result is not good, and this is mainly because of long sequence length of this set. VDGA decomposes alignments into sub alignments and aligns sub parts separately; therefore length of input sequences cannot have a strong effect on the performance of this algorithm. In 1ad2 which has sequences with medium size, the same problem may result in lower efficiency of our method. But in this set still new algorithm performs better than MSA-GA and ClustalW. Although VDGA_Decomp uses guide trees and also its powerful decomposition method, our simple method still shows comparable results with that.

4.2. *Compare matched columns with AlineaGA and T-COFFEE*

The purpose of this experiment is to compare the proposed algorithm with AlineaGA[14] and also comparison with T-COFFEE given in AlineaGA.[14] We choose AlineaGA which was a new genetic algorithm for solving multiple sequence alignment. AlineaGA[14] is an improvement of previous version of AlineaGA in Ref. 21. There are several mutation and crossover operation defined in AlineaGA. Crossover operator used in AlineaGA are single-point crossover (like SAGA crossover with linkage problem), and recinbineMatchCols[19] that combines two genes trying to keep matched columns of both genes. Mutation operators are gap insertion, smart gap insertion, gap shifting, smart gap shifting, merge space, smart merge space, and columnGapRemover. These are some operators, some with smart versions of them which uses local search in order to perform the mutation or cross over in a more effective way.

It is obvious that some of these operations are biased toward finding fully matched columns. The fitness function used in AlineaGA is also a weighted sum of pairs score and number of fully matched columns. We could not find a proof for usefulness of considering matched columns but lots of methods have took it into consideration; but

in an intuitive point of view it seems that it helps to find more matched areas and therefore better alignments. On the other hand, not considering matched columns score in the objective function may leads to another problem: consider the situation that operators try very hard for forming matched columns and because the objective function does not care about it, improved genes will not be considered as good genes. This approach can help if we know that having fully matched columns really improve the result. In our experiments, there were some cases that considering number of fully matched columns in objective function made the results even worst; because finding fully matched columns was acting in a greedy way that destroyed other parts of the alignment. We cannot say certain statements about this, but there is a need for studying the effect of this factor in a scientific way.

In Ref. 14, four sets of sequences have been selected, and SPS score, number of matched columns and number of inserted gaps are compared to the progressive T-COFFEE method. It should be mentioned here that because the operators and objective function are biased toward finding fully matched columns, it cannot be a fair factor for comparison; anyway we compared our method results on the same set of sequences on both sum of pairs score and number of matched columns. Sum of pairs score comparison was omitted here because the information about gap penalty was not given in the paper. We found some factors leading to the same result but because of not being sure, the result is omitted here. AlineaGA is acting very greedy for finding fully matched columns so it is possible that it inserts lots of gaps into the sequences, therefore the designers of algorithms have put the number of inserted gaps as an input parameter for the algorithm. Tables 3 and 4 compares our result in number of fully matched columns and number of inserted gaps with Ref. 14.

AlineaGA uses three different configurations for each test. In each configuration the number of allowed gap insertion, the probability of each operation and other genetic algorithm parameter setting is specified (Table 1 of Ref. 14 contains the parameter setting for each configuration). Comparing the results given in Tables 3 and 4 shows that although our method is very simple and does not consider matched

Table 3. Comparing number of matched columns in new method with AlineaGA on the sequence set 1, 2 and 3 of Ref. 14.

Sequence Set	AlineaGA Config	Average AlineaGA	Best AlineaGA	T-COFFEE	Average New Method	Best New Method
1	1	65.3	65	61	64.1	65
	2	64.9	65			
	3	64.7	65			
2	1	42.9	48	46	40.4	48
	2	43.1	47			
	3	43.2	49			
3	1	29.9	32	39	30.8	36
	2	29.2	37			
	3	30.4	36			

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

Table 4. Comparing number of inserted gaps in new method with AlineaGA on the sequence set 1, 2 and 3 of Ref. 14.

Sequence Set	AlineaGA Config	Best AlineaGA	T-COFFEE	Best New Method
1	1	9	9	9
	2	9		
	3	9		
2	1	20	20	20
	2	20		
	3	36		
3	1	46	29	77
	2	41		
	3	41		

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

columns in the objective function, its results are comparable to a complicated method like AlineaGA.

5. Conclusion

In this paper, we proposed a simple genetic algorithm for multiple sequence alignment. The proposed algorithm uses simple, random and not problem specific operators and this makes our algorithm very fast in contrast to existing genetic algorithm. We showed through simulation that it is not necessary to use complicated operators for complicated problems. The proposed operators consist of simple mutation operators, a recombination operator and a new operator called cross-mutation. Cross-mutation is in fact a new way for better vertical recombination of two genes which does not have the problem of biased cut points and minimizes the linkage problem. Another problem in existing methods was about their population initialization. We proposed a new method for population initialization that has the advantage of randomness and using initial seeds. The advantage of this method is that it does not use initial seed directly into its first population; hence it does not have the problem of getting entrapped in the local optima of seed. We also used ordinary fitness function (sum of pairs score plus gap penalty), which is a suitable objective function from biological point of view too. The experimental results show the suitableness of this objective function in contrast to other objective functions. The new method showed better efficiency on shorter sequences. For future work we can use our new operators beside a vertical decomposition technique to solve the problem of large sequences.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which improved the paper.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com by 88.198.134.171 on 02/04/13. For personal use only.

References

1. P. Clote and R. Backofen, *Computational Molecular Biology: An Introduction* (John Wiley & Sons, 2000).
2. C. Gondro and B. P. Kinghorn, A simple genetic algorithm for multiple sequence alignment, *Genetics Molecular Res.* **6**(4) (2007) 964−982.
3. R. Thomsen, G. B. Fogel and T. Krink, A clustal alignment improver using evolutionary algorithms, in *Proc. Fourth Congress on Evolutionary Computation* (*ECE-2002*), Vol. 1, (2002), pp. 121−126.
4. C. Notredame and D. G. Higgins, SAGA: Sequence alignment by genetic algorithm, *Nucleic Acids Res.* **24**(8) (1996) 1515−1524.
5. G. J. Barton and M. J. E. Sternberg, A strategy for the rapid multiple alignment of protein sequences, *J. Molecular Biol.* **198** (1987) 327−337.
6. W. Taylor, A flexible method to align large numbers of biological sequences, *J. Molecular Biol.* **28** (1988) 61−169.
7. J. Devereux, P. Haeberli and O. Smithies, A comprehensive set of sequence analysis programs for the VAX, *Nucleic Acids Res.* **12** (1984) 387−395.
8. J. D. Thompson, D. G. Higgins and T. J. Gibson, CLUSTALW: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Res.* **22** (1994) 4673−4680.
9. R. C. Edgar, MUSCLE: Multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Res.* **32**(5) (2004) 1792−1797.
10. C. Notredame, D. Higgins and J. Heringa, T-Coffee: A novel method for multiple sequence alignments, *J. Molecular Biol.* **302** (2000) 205−217.
11. N. Saitou and M. Nei, The neighbor-joining method: A new method for reconstructing phylogenetic trees, *Molecular Biol. Evol.* **4** (1987) 406−425.
12. P. H. Sneath and R. R. Sokal, *Numerical Taxonomy* (Freeman, San Francisco, 1973), pp. 188−308.
13. L. Brocchieri and S. Karlin, Asymmetric-iterated multiple alignment of protein sequences, *J. Molecular Biol.* **276** (1998) 249−264.
14. F. J. M. Silva, J. M. S. Pérez, J. A. G. Pulido and M. A. V. Rodríguez, AlineaGA — a genetic algorithm with local search optimization for multiple sequence alignment, *Appl. Intell.* **32** (2010) 164−172.
15. F. Naznin, R. Sarker and D. Essam, Vertical decomposition with genetic algorithm for multiple sequence alignment, *BMC Bioinf.* **12** (2011) 353.
16. J. Pei and N. V. Grishin, MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information, *Nucleic Acids Res.* **34**(16) (2006) 4364−4374.
17. R. Roshan and D. R. Livesay, Probalign: Multiple sequence alignment using partition function posterior probabilities, *J. Bioinf.* **22**(22) (2006) 2715−2721.
18. A. Bahr, J. D. Thompson, J. C. Thierry and O. Poch, BAliBASE (Benchmark Alignment dataBASE): Enhancements for repeats, transmembrane sequences and circular permutations, *Nucleic Acids Res.* **29**(1) (2001) 323−326.
19. K. Chellapilla and G. B. Fogel, Multiple sequence alignment using evolutionary programming, in *Proc. First Congress of Evolutionary Computation* (*CEC-1999*), (1999), pp. 445−452.
20. V. A. Simossis and J. Heringa, PRALINE: A multiple sequence alignment toolbox that integrates homology-extended and secondary structure information, *Nucleic Acids Res.* **33** (2005) 289−294.
21. F. J. M. Silva, J. M. S. Pérez, J. A. G. Pulido and M. A. V. Rodríguez, AlineaGA: A genetic algorithm for multiple sequence alignment, in *New Challenges in Applied*

Intelligence Technologies, eds. N. T. Nguyen and R. Katarzyniak (Springer, Berlin, 2008), pp. 309−318.
22. 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. Molecular Biol.* **48** (1970) 443−453.
23. M. Kimura, A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences, *J. alaMolecular Evol.* **16** (1980) 111−120.
24. D. Thierens and D. E. Goldberg, Mixing in genetic algorithms, in *Proc. Fifth Int. Conf. Genetic Algorithms* (1993), pp. 38−45.
25. R. Thomsen and W. Boomsma, Multiple sequence alignment using SAGA: Investigating the effects of operator scheduling, population seeding, and crossover operators, *Appl. Evol. Comput.* **3005** (2004) 113−122.
26. M. Pelikan, D. E. Goldberg and E. Cantu'-Paz, Linkage problem, distribution estimation, and bayesian networks, *Evol. Comput.* **8**(3) (2000) 311−340.
27. The UniProt Consortium, The universal protein resource (UniProt), *Nucleic Acids Res.* **36** (2008) 190−195.

Int. J. Comp. Intel. Appl. Downloaded from www.worldscientific.com
by 88.198.134.171 on 02/04/13. For personal use only.