Journal of Advanced Computer Science & Technology, 5 (2) (2016) 28-33

Journal of Advanced Computer Science & Technology

Website: www.sciencepubco.com/index.php/JACST
doi: 10.14419/jacst.v5i2.6079
Research paper

Survey of the use of genetic algorithm for multiple sequence alignment

Mohamed Tahar Ben Othman *

Senior Member, IEEE, Computer Science Dept., College of Computer,
Qassim University, Kingdom of Saudi Arabia
**Corresponding author E-mails: maathaman@qu.edu.sa, mtothman@gmail.com*

Abstract

Multiple Sequence Alignment (MSA) is used in genomic analysis, such as the identification of conserved sequence motifs, the estimation of evolutionary divergence between sequences, and the genes' historical relationships inference. Several researches were conducted to determine the level of similarity of a set of sequences. Due to the problem of the NP-complete class property, a number of researches use genetic algorithms (GA) to find a solution to the multiple sequence alignment. However, the nature of genetic algorithms makes the complexity extremely high due to the redundancy provided by the different operators. The aim of this paper is to study some proposed GA solutions provided for MSA and to compare them using some criteria which we believe any solution should comply with in matters of representativeness, closeness and original sequence invariance.

Keywords: *Genetic Algorithms; Multiple Sequence Alignment; Representation Closeness; Representativeness; Sequence Invariance.*

1. Introduction

Genetic algorithms are search heuristics which mimic the natural selection process. They are used to generate good solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA). They generate solutions to the optimization problems using techniques inspired from natural evolution, such as inheritance, mutation, selection, and crossover. The general GA algorithm is described in Fig. 1:

1) Randomly generate the first population of n chromosomes
2) Evaluate the fitness function for each chromosome
3) Repeat the following steps to create new population
4) Select two parent chromosomes to crossover (with the crossover rate) and generate offspring(s)
5) offspring(s) mutation (with the mutation rate)
6) Evaluate the fitness of the offspring(s)
7) Select chromosomes for discard (this can be done at the each new-born level, (offspring(s) inside the loop within a population or at the population level outside the loop)

Fig. 1: Generic Genetic Algorithm.

The algorithm generally terminates either by reaching a chosen number of generations or by attaining the stability of the best-found fitness. The fitness stability can be caused by a local optimum.
There are several parameters that should be chosen before starting this algorithm: beginning with the chromosome representation, then going through the process of choosing the number of chromosomes in the first population, completing a fitness calculation, initiating a parent selection, defining the crossover operator and rate, the mutation operator and rate, and the number of offspring(s), activating the discard technique and ending with deciding the way of the algorithm termination.
The aim of this paper is to focus on these different GA parameters used in MSA through a set of researchers. We will use the same randomly generated DNA sequences given in Fig. 2 throughout this paper:

S_1	TTATGACGTT
S_2	ATTCTACTTT
S_3	GATTGTGCGA
S_4	GACAATGCTA

Fig. 2: Used Set of Sequences.

2. Chromosome representation

The chromosome representation is mainly chosen to:

a) Adapt to the optimization problem (here MSA) to GA operators
b) Simplify the alignment so that most of the operations will be done on the representation and not on the aligned sequences.
c) Optimize the processing and reduce the complexity

To be able to achieve these objectives, we believe that the proposed solution should comply with the following rules:

a) Closeness: Any operation should be close to the representation space in a way that after the operation is performed on a chromosome, the result should remain within the criteria set for the representation. Otherwise, the process may be left with some chromosomes that can be either truncated or not be represented at all, which drifts away from the process' goal.
b) Representativeness: A representation should be able to represent any possible alignment even in a reduced space (i.e. by fixing the maximum number of gaps).
c) Sequence Invariance: An operation on a representation should not damage any original sequence. Otherwise, the

alignment solution may not be a solution of the original sequences.

Besides these rules, the main issue in genetic algorithms is the considerable time of execution. The condition to reduce the execution time is to have genetic algorithm operators that lead to convergence, which is difficult to achieve considering the nature of these kind of functions. Researchers are attempting to have this convergence by selecting the best fitness; and the trigger to stop the execution is either a maximum number of populations or a threshold level of fitness value set as parameters. Both of these choices may not lead to an accepted solution.

There are several chromosome representations used in different research papers among which hose discussed below.

2.1. Bit matrix and quantum representation

A chromosome is a NxM bit matrix [3] or Quantum Representation [4]. As showed in Fig. 3, a sequence, including gaps, in an alignment is represented as a bit string. In this bit string, `1' (or '0') corresponds to a gap, and the total number of `0's (or '1's) represents the length of the sequence. The alignment is expressed as a matrix, which is a vertical arrangement of the bit strings.

0 0 1 0 1 1 0 1 1 0 0 0 0 0 0	TT-A--T--GACGTT
0 1 1 0 1 0 1 0 0 0 0 0 0 0 0	A--T-T-C-TACTTT
1 0 0 0 0 1 0 1 0 0 0 0 0 0 0	-GATT-G-TGCGA
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0	GACAATGC-TA
Chromosome	Correspondent Alignement

Fig. 3: Bit String Representation.

This representation can be used to implement the alignment algorithm using hardware to reduce the execution time, but it does not take into account the different nucleotides which must be managed in parallel, and thus contrary to the chromosome representation goal.

2.2. Steady GA

A steady GA is used in [7] that represents a chromosome with an N*M matrix (N is the number of sequences and M is the size of the longest sequence extended by 30% of gaps). Each row i describes the gap positions in i^{th} sequence as presented in Fig. 4.

3 5 6 8 9 0 0 0 0 0 0 0 0 0 0	TT-A--T--GACGTT
2 3 5 7 9 0 0 0 0 0 0 0 0 0 0	A--T-T-C-TACTTT
1 6 8 0 0 0 0 0 0 0 0 0 0 0 0	-GATT-G-TGCGA
9 0 0 0 0 0 0 0 0 0 0 0 0 0 0	GACAATGC-TA
Chromosome	Correspondent Alignement

Fig. 4: Gaps Indexes Representation.

2.3. Permutation representation

A permutation solution PS in [13] is associated with each alignment solution. PS is a matrix MxN for M sequence with the total size, including gaps, represented by N. Each row represents the indexes of their elements in the alignment. If the original sequence size is n, then the first n elements of the row are the new indexes of the original sequence elements and the remaining represent the gaps positions. Fig. 5 provides an example:

1 2 4 7 10 11 12 13 14 15 3 5 6 8 9	TT-A--T--GACGTT
1 4 6 8 10 11 12 13 14 15 2 3 5 7 9	A--T-T-C-TACTTT
2 3 4 5 7 9 10 11 12 13 14 15 1 6 8	-GATT-G-TGCGA
1 2 3 4 5 6 7 8 10 11 12 13 14 15 9	GACAATGC-TA
Chromosome	Correspondent Alignment

Fig. 5: Permutation Representation.

Both representations in 2.2 and 2.3 use the sequences indexing. In the first, the focus is in only the gaps indexes, whereas in the second, all sequences including gaps indexes are used. For the same chromosome, the first representation is included at the end of the second representation. Although the 2-2 representation is more concise and any different permutation of the gaps' indexes in the 2-3 is representing the same sequence, the GA functions over a permutation is easier.

2.4. Adaptive genetic algorithm

In [8] there is no particular representation of the chromosome as the sequences themselves are used directly which does not reduce the complexity of sequences management.

2.5. Divide and conquer

This method is used in [10] wherein each DNA gene is represented by two bits as shown in Table 1:

Table 1: DNA Coding

DNA Data	DNA Symbol	New Format
Adenine	A	00
Cytosine	C	01
Guanine	G	10
Thymine	T	11

The chromosome is then a set of binary strings. Fig. 6 shows an example of using this representation:

S_1	TTATGACGTT	11110011100001101111
S_2	ATTCTACTTT	00111101110001111111
S_3	GATTGTGCGA	10001111101110011000
S_4	GACAATGCTA	10000100001110011100

Fig. 6: Bit Coding Representation.

The chromosome representation is mainly used to ease the GA operators by reducing the overall complexity of this algorithm. What distinguishes this technique is that all GA operations should be done at the gene level and not at bit level as it is done in [10]. Crossover and mutation that were presented in the paper are executed in bit level which may modify the main DNA original sequences. Moreover, there is no gap representation. Table 2 gives a comparison between all studied representation techniques vs. the performance criteria set described at the beginning of this paper.

Table 2: Representations vs. Performance Criteria

Representation	**Performance Criteria**		
	Closeness	Representativeness	Invariance
Quantum Representation	This Representation Aims To Simplify The Processing Through Parallelism And/Or By Building A Hardware Solution. Certainly, If The Operations Are Respecting Even Bit Boundary, The Result Of Falls Into The Representation Space.	The Gap Is Not And Cannot Be Represented As Only Two Bits Are Used.	Some Proposed Solutions While Processing At A Bit Level Do Not Specify How To Keep The Original Sequences Invariant. If The Operations Are Completed At The Bit Level It Certainly Impacts The Invariance. For Example In [10] The Different Genes Are Represented With Two Bits, If An Operation Is Done At The Bit Level A Gene 'A' May Change To 'C' By Changing The First Bit.
Divide And Conquer			

Bit Matrix	In The Proposed Representation In [3] A Bit Is A Gap Or A Gene. The Major Problem In This Solution Is A Ga Operator (I.E. Crossover) May Include More Gaps Than Supported And Then Part Of The Sequence Will Not Be Represented.	This Solution Is Generally Not Representative As A Static Matrix Cannot Hold All Chromosomes Unless A Maximum Number Of Gaps Is Allowed, Which Will Add More Complexity In Tracking The Number.	The Representation, While It Is Respecting The Rules 1 And 2, Does Not Impact The Original Sequences.
Steady Ga	Although, It May Be Easily Managed, The Number Of Gaps Are Either Maximized, Which Complicates Its Management, Or - For More Flexibility - It Is Not And An Operation Adding Gaps May Conclude With The Total Matrix Not Close To The Predefined Space. Also, All Numbers Representing The Gaps Indexes Should Be Maintained Distinct.	Both Solutions Are Representative If The Number Of Gaps Is Delimited. While There Is More Flexibility In The First Solution, There Is No Need For Tracking In The Second, Which Reduces The Complexity.	With The Need For Extra-Management In The Steady Solution When Gaps Flexibility Is Used, Both Solutions Do Not Impact The Original Sequences.
Permutation Representation	Although Different Permutations May Represent The Same Chromosome, Without Any Complexity Increase, All Ga Operation Is Close To This Representation.		
Adaptive Genetic Algorithm	As seen in [8], the authors are using the sequences themselves and not a transformation. All rules are respected at the cost of the processing complexity.		

3. Gap insertion

Symbolized by the '-' character in alignments, gaps are used to align the sequences. They represent the insertion or deletion of a gene, or genes, in one of the genetic sequences. They give the ability to account the addition or missing information between aligned sequences. It is comparable to the problem of missing information in deteriorated or damaged papyrus fragments [9]. Gaps are inserted in sequences so that the same genes in different sequences will be on top of each other as frequently as possible. Each gap introduces a penalty. Some papers [1] use different methods of scoring gaps: opening, extending and terminal gaps, which may have different penalties.

3.1. First generation

Most of the research papers create the first generation randomly, depending on the chromosome representation. Random means that the gaps are randomly inserted in the different sequences. The papers that penalize differently gaps generally give the lowest penalty for the first population inserted gaps.

3.2. Fitness function

Most of the papers use the Sum of Pair Method, which is the sum of fitness values between all pairs of sequences. The main difference is how to calculate the similarity (fitness) of two sequences.

3.3. Weighted sum of pair method (WSPM)

$$\text{Alignment cost(A)} = \sum_{i=2}^{N}\sum_{j=1}^{i-1} W_{i,j}\ \text{cost}(A_i, A_j) \quad (1)$$

Where cost is the score of the similarity between two sequences (A_i and A_j) and $W_{i,j}$ is their weight.

3.4. Score coffee

The score COFFEE described in the equation 2 is used in [4].

$$\text{Alignment cost(A)} = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{i,j}\text{cost}(A_i,A_j)}{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} W_{i,j} * \text{Len}} \quad (2)$$

3.5. Gap penalty distinction

To impose a start-up penalty for new gaps a distinction between "gap groups" and "individual gaps" is introduced in [2]. The fitness in this work is calculated by adding 1 for each matching pair of symbols, and subtracting 4 for every group of consecutive gaps, and 0.4 for each individual gap.

fitness = (total matches) * 1.0 - (gap penalties) (3)

gap penalties = (gap groups) * 4.0 + (total number of gaps) * 0.4

In [13] there is no difference between the different gaps, but there is a penalty of -1 if there is no match between genes and of -2 if there are a gene and a gap. The gain of a match is 2.

3.6. Score matrix

Most algorithms define a score matrix to define the cost (A_i, A_j) for all A_i, $A_j \in \{A, C, G, T, -\}$:

Table 3: Score Matrix Used in [8]

	A	C	T	G	-
A	0	5	2	5	10
C	5	0	5	2	10
T	2	5	0	5	10
G	5	2	5	0	10
-	10	10	10	10	0

Table 4: Score Matrix Used in [12]

	A	C	T	G	-
A	1	-1	-1	-1	-2
C	-1	1	-1	-1	-2
T	-1	-1	1	-1	-2
G	-1	-1	-1	1	-2
-	-2	-2	-2	-2	0

The values in the score matrix do not change for all the processing time except for those penalizing the gaps differently. Also, most papers do not give biological reasons behind the values they propose, as seen in Table 3 and Table 4, barring those using the Point Accepted Mutation, also known as PAM [11] or the Blocks Substitution Matrix (also called BLOSUM matrix) [9]. PAM describes the replacement of an amino acid in the primary structure of a protein with another amino acid. There are several PAM versions. The most used is pam250. The score matrix is used to calculate the fitness, which has to be either minimized when it describes the penalty, as shown in table 1, or maximized when it describes the similarity as demonstrated in table 2. The Blocks Substitution Matrix (also called BLOSUM matrix) [9] - based on conserved blocks bounded in similarity - was calculated by extracting sections of alignment from a database of observed genetic

sequence alignments. Once the relative frequencies for each amino acid were calculated, a log-odds ratio was recorded for every possible amino acid substitution pair. The formula for constructing the BLOSUM matrix is:

$$S_{ij} = \frac{1}{\lambda}\log\left(\frac{p_{ij}}{q_i q_j}\right) \quad (4)$$

Where p_{ij} is the probability of two amino acids i and j replacing one another in any sequence and q_i is the background frequency of finding amino acid i in any sequence. λ is a scaling factor.

4. Selection technique

In Genetic Algorithm, the selection mechanism is a process that aims to recruit the better individuals for the next generation. The selection technique provides a way to selectively favor the better individuals. The selection technique is used to select the parent chromosomes for crossover and mutation, which produces offspring children. Several selection techniques are used:

4.1. Random

Two parents are randomly selected each time for crossover. This technique is simple, but cannot always lead to the most beneficial results.

4.2. Best fitness

This technique needs a sorted population, which adds complexity, believing that "the best parents potentially give better offspring," which is not always true.

4.3. Tournament selection

Each time a number of individuals (called the tournament size) is chosen from the population at random. For the crossover, from the pool/tournament choose the best individual as the first parent with probability p and choose the second best individual as the second parent with probability p*(1-p). If p=1, the best fitness individual will be chosen.

4.4. Proportional selection

There are different algorithms for proportional selection. The most popular are:

- Roulette Wheel Selection (RWS),
- Stochastic Reminder Roulette Wheel Selection (SRRWS), and
- Stochastic Universal Sampling (SUS).

In this technique, a probability of selection is associated with each chromosome. The probability of a chromosome i with a fitness f_i is calculated using the equation:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \text{where N is the total number of chromosomes} \quad (5)$$

4.5. Ranking selection

The Ranking selection is identical to the proportional selection; however, it commences by ranking the chromosomes using their fitness, which helps to avoid premature convergence.

5. Crossover operation

Several types of crossover operators are proposed in the research field. This operator is based on an analogy with biological crossover:

5.1. One-point crossover

The first parent is cut straight at some randomly chosen position and the second one is tailored so that both right and left pieces of each parent can be joined while respecting the invariance rule of the original sequences. Any void space that appears at the junction point is filled with gaps. This technique is used in [1]. An example is given in Fig. 7:

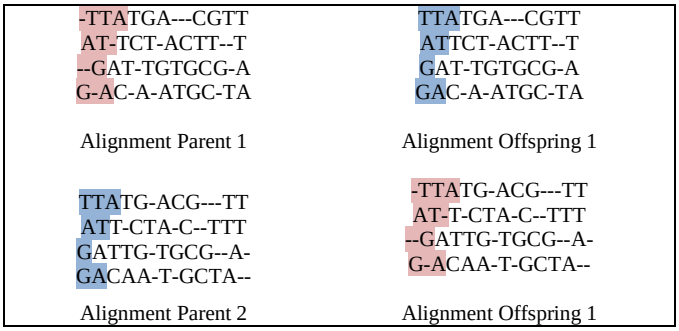

Fig. 7: One-Point Crossover.

5.2. Uniform crossover

Promotes multiple exchanges between two parents at gene level rather than the segment level. The algorithm consists of two main steps. Step 1 seeks to find the consistent positions in the parents' alignments. Step 2 exchanges the subsequences between two positions with a probability typically equal to 0.5. Two positions are considered consistent between two alignments if, in each row, they contain the same residue or a gap. The main flaw of this technique is that it fails in most cases to satisfy the invariance rule. In [1], a position is a column of residues or gaps in an alignment. This technique may work mainly for sequences with high similarity. To preserve the sequence invariance, the definition of consistency given in [1] should be extended to the fact that all genes before the position should be the same in the same rows in the parent alignments. On the other hand, forcing column (same position in all sequences) reduces the number of possibilities failing the invariance rule.

5.3. Window-frame crossover

The window-frame crossover is used in [3]. Some windows are selected in each parent, and they are copied in the same sequences in a copy of the second parent. The main setback of this technique is that it fails the representativeness rule when the number of gaps exceeds a certain threshold. Fig. 8 presents an example of a window-frame crossover:

As demonstrated, it cannot preserve the maximum number of gaps that are used in a sequence, which may lead to the lack of representation of some gaps or genes in the chromosome; or they can be simply considered dummies (they exist but with no effect). The result of such alignment cannot be considered as nearly optimal as not all possible alignments can be represented.

5.4. Partially matched crossover (PMX)

Let p_1 and p_2 represent parent chromosomes in permutations space. Two random numbers between 1 and the length n are generated and set as Lower-Level (LL) and Upper-Level (UL) as showed in Fig. 9. The segment of p_1 between LL and UL is copied to form a partial list of offspring po in the same position as it appears in p_1. All the remaining positions in po are copied in order from p_2. This technique is used in [13].

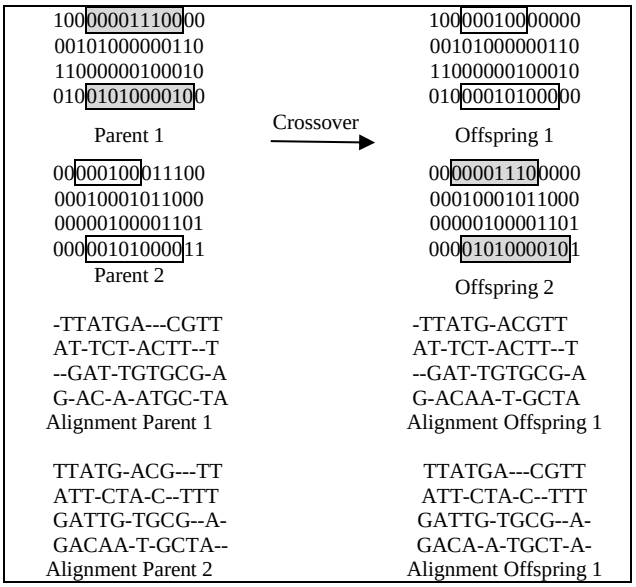

Fig. 8: The Window-Frame Crossover.

P1	3	9	5	4	6	10	7	8	1	2
P2	10	4	5	2	9	7	3	6	1	8
					LL=3, UL=7					
P3	2	9	5	4	6	10	7	3	1	8

Fig. 9: Crossover Illustration.

5.5. Position based crossover (PBX)

The algorithm of the PBX [13] can be summarized as follows:

- Select randomly a set of position from one permutation.
- Produce a proto-permutation child by copying genes (elements) on these positions into the corresponding position of the proto-permutation child.
- Delete genes, which are already selected from the second permutation. The resulting sequence of permutations elements contains the elements of the proto-permutation child's needs.
- Place the chromosomes (permutations) into the unfixed position of the proto child from left to right according to the order of the sequence to produce one offspring (permutation). Fig. 10 gives an illustration of PBX crossover:

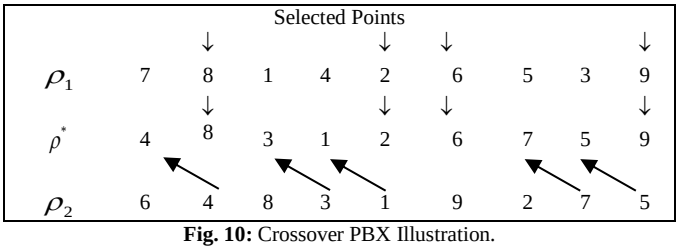

Fig. 10: Crossover PBX Illustration.

5.6. Cycle crossover (CX)

Fig. 11 presents an illustration of CX crossover [13]. Starting from a position in the permutation of one sequence and copying the index at the same position in the offspring, the next element from the same sequence has the index with the value having the same position as the first in the second sequence. This process is repeated until a cycle is found (return to the starting point). All remaining values are copied from the permutation of the second sequence.

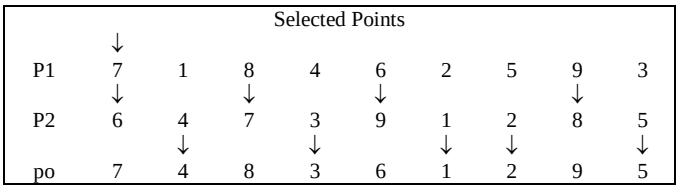

Fig. 11: Crossover CX Illustration.

5.7. Hybrid crossover (HX)

The Hybrid Crossover aims to randomly use one of the sets of different crossovers operators [13].

6. Mutation operation

6.1. Delimited position mutation

In this mutation, a delimited position (dp) is selected for the permutation [13], as shown in Fig. 12. This position divides the permutation into two parts from which two alleles are randomly selected and exchanged.

P	2	4	5	8	6	10	7	3	9	1
					dp=4					
P_m	2	4	7	8	6	10	5	3	9	1

Fig. 12: Delimited Position Mutation.

6.2. Island shift mutation

The island shift mutation is used in [3]. Some windows, called islands, are selected in some sequences as presented in Fig. 13. The mutation point is selected either at the beginning or at the end of the island and the mutation is done by a shift between this point and the opposite.

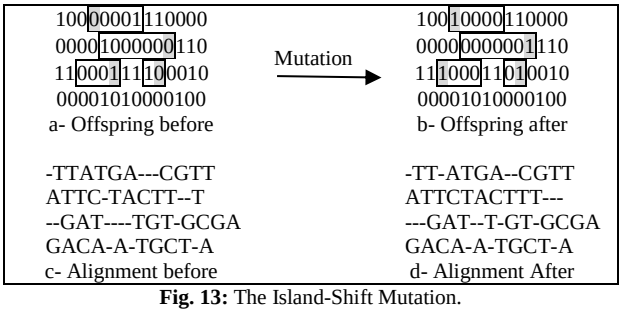

Fig. 13: The Island-Shift Mutation.

6.3. Mutation operations discussion

We start with the last kind of mutation operation namely "Island shift mutation" on a binary chromosome representation, where '1' represents a gap. As aforementioned in the crossover using this kind of chromosome representation, the mutation may lead to inconsistency and the failing representativeness rule (as we can see in the alignment in Fig. 13-d compared to the alignment representation in Fig. 13-b). The size of the array is static, and then when doing the mutation operation, we may insert more gaps than authorized or replace some 0's with 1's, which may lead to some genes or gaps not being represented on the chromosome representation. An example is given in the sequence 2, where the last zero in the offspring after the mutation holds no significance. Also in sequence 3, the last three genes are not represented in the matrix and no possibility is given to insert a gap in them, which means that the proposed solution may not lead to a good result and it cannot cover all of them.

7. Cross-over and mutation probabilities

Although most of the research papers use a probability of one (1) for crossover and mutation operations, some include probabilities such as:

$$P_m = \frac{-1}{1+e^{-k2*\Delta}} + 1.0, \quad \Delta = favg - fmax \qquad (6)$$

(favg, (resp fmax)is the average (resp max)fitness value of a population)

8. Discard techniques

Most of the papers use the same population size over the time of the execution. To maintain this size, a discard technique should be periodically executed. This period can be between every two populations or after the arrival of each new offspring resulting from a crossover and mutation operations. There are several techniques among which a) discarding those having poor fitness, b) discarding randomly, are options. In some papers, a hybrid solution is used where only the better of the two children is kept [1] and between two populations a global discard is done. The latter has an effect only if the new offspring is active in its birth population, which generally is not the case unless the selection technique is random.

9. Conclusion

The aim of this study is to validate different genetic algorithm operators used for multiple sequence alignment against the rules of representativeness, closeness, and invariance. Although, Bit matrix, Adaptive Genetic Algorithm, and Permutation representation have full representativeness, only the latter two meet the closeness and original sequence invariance. The difference of complexity of the execution depends mainly on the way the fitness is calculated. More precisely, it depends on the reduction of the number of original sequences comparisons. This is more thoroughly dissected in [13, 17] using the Permutation representation. The main common setback for all mentioned solutions, to the exception of those using hardware, is how to ensure the convergence by reducing the work space each time. This can be achieved by reducing the redundancy provided by the nature of genetic algorithms, which will be focused on in the future work.

References

[1] C. Notredame, D.G. Higgins, "SAGA: *Sequence Alignment* by *Genetic Algorithm*", Nucleic Acid Research, Vol. 24, 1515-1524, 1996 http://dx.doi.org/10.1093/nar/24.8.1515.

[2] Kosmas Karadimitrion and Donald H. Kraft, "Genetic Algorithms and The Multiple Sequence Alignment Problem in Biology", Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, Baton Rouge, LA., February 1996.

[3] Isokawa M, Wayama M, Shimizu T, "Multiple sequence alignment using a genetic algorithm", Genome Informatics, 1996, 7:176–177.

[4] Layeb, A. ; Meshoul, S. ; Batouche, M., "Multiple sequence alignment by quantum genetic algorithm", 20th International Parallel and Distributed Processing Symposium (IPDPS), 2006, http://dx.doi.org/10.1109/IPDPS.2006.1639617.

[5] Naznin F, Sarker R, Essam D. "Vertical decomposition with Genetic Algorithm for Multiple Sequence Alignment.", BMC Bioinformatics 2011; 12:353; PMID:21867510; http://dx.doi.org/10.1186/1471- 2105-12-353.

[6] Naznin, F. ; Sarker , R. ; Essam, D., "Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment", IEEE Transactions on Evolutionary Computation, Vol 16, Issue: 5, http://dx.doi.org/10.1109/TEVC.2011.2162849.

[7] Pramanik, S. ; Setua, S.K., "A steady state Genetic Algorithm for Multiple Sequence Alignment", International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, http://dx.doi.org/10.1109/ICACCI.2014.6968251.

[8] Liu Chao ; Liu Shuai, "The research on DNA multiple sequence alignment based on adaptive immune genetic algorithm", International Conference on Electronics and Optoelectronics (ICEOE) , 2011, Vol 3, http://dx.doi.org/10.1109/ICEOE.2011.6013304.

[9] Williams, A.C. ; Carroll, H.D. ; Wallin, J.F. ; Brusuelas, J. ; Fortson, L. ; Lamblin, A.- F. ; Haoyu Yu, "Identification of Ancient Greek Papyrus Fragments Using Genetic Sequence Alignment Algorithms", IEEE 10th International Conference on e- Science (e-Science), 2014, Vol 2, http://dx.doi.org/10.1109/eScience.2014.14.

[10] Al Junid, S.A.M. ; Reffin, M.S. ; Majid, Z.A. ; Tahir , N.M. ; Har on, M.A., "Implementation of genetic algor ithm for optimizing DNA sequence alignment", IEEE Business Engineering and Industrial Applications Colloquium (BEIAC) , 2012. http://dx.doi.org/10.1109/BEIAC.2012.6226111.

[11] Naznin, F. ; Sarker , R. ; Essam, D., "DGA: Decomposition with genetic algorithm for multiple sequence alignment", IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) , 2010, http://dx.doi.org/10.1109/CIBCB.2010.5510595.

[12] Miranda, L.A. ; Caetano, M.A.F. ; Melo, A.C.M.A. ; Correa, J.M. ; Bordim, J.L., "Multiple Biological Sequence Alignment with a Parallel Island Injection Genetic Algorithm", 12th IEEE International Conference on High Performance Computing and Communications (HPCC) , 2010, http://dx.doi.org/10.1109/HPCC.2010.31.

[13] Ben Othman, M.T. ; Abdel- Azim, G., "Multiple sequence alignment based on genetic algorithms with new chromosomes representation", Electro-technical Conference (MELECON) , 2012 16th IEEE Mediterranean, http://dx.doi.org/10.1109/MELCON.2012.6196603.

[14] Kumar S, Filipski A (2007) Multiple sequence alignment: In pursuit of homologous DNA positions. Genome Res 17: 127–135. http://dx.doi.org/10.1101/gr.5232407.

[15] T. Manning, R.D. Sleator, P. Walsh, Naturally selecting solutions: The use of genetic algorithms in bioinformatics, Bioengineered 4 (2012) 266–278. http://dx.doi.org/10.4161/bioe.23041.

[16] Amouda Nizam, Jeyakodi Ravi, and Kuppuswami Subburaya, "Cyclic Genetic Algorithm for Multiple Sequence Alignment", International Journal of Research and Reviews in Electrical and Computer Engineering (IJRRECE), Vol. 1, No. 2, June 2011.

[17] Mohamed Tahar Ben Othman, Gamil Abdel-Azim, "Genetic Algorithms with Permutation Coding for Multiple Sequence Alignment.", Recent Patent DNA Gene Seq. 2013: 22974260, VOLUME: 7, ISSUE: 2, Page: [105 - 114], Pages: 10, http://dx.doi.org/10.2174/1872215611307020004.