Global Journal of Computer Science and Technology
Volume 11 Issue 7 Version 1.0 May 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
ISSN: 0975-4172 & Print ISSN: 0975-4350

Self-Organizing Genetic Algorithm for Multiple Sequence Alignment

By Amouda Nizam, Buvaneswari Shanmugham, Kuppuswami Subburaya
Pondicherry University

Abstract- Genetic algorithm (GA) used to solve the optimization problem is self-organized and applied to Multiple Sequence Alignment (MSA), an essential process in molecular sequence analysis. This paper presents the first attempt in applying Self-Organizing Genetic Algorithm for MSA. Self-organizing genetic algorithm (SOGA) can be developed with the complete knowledge about the problem and its parameters. In SOGA, values of various parameters are decided based on the problem and fitness value obtained in each generation. The proposed algorithm undergoes a self-organizing crossover operation by selecting an appropriate rate or a point and a self-organizing cyclic mutation for the required number of generations. The advantages of the proposed algorithm are (i) reduce the time requirement for optimizing the parameter values (ii) prevent execution with default values (iii) avoid premature convergence by the cyclic mutation operation. To validate the efficiency, SOGA is applied to MSA, and the resulting alignment is evaluated using the column score (CS). The comparison result shows that the alignment produced by SOGA is better than the widely used tools like Dialign and Multalin. It is also evident that the proposed algorithm can produce optimal or closer-to-optimal alignment compared to tools like ClustalW, Mafft, Dialign and Multalin.

Keywords: *Crossover, Genetic Algorithm, Multiple Seuence Alignment, Mutation, Selection, Self organization*

GJCST Classification: *J.3*

Strictly as per the compliance and regulations of:

Self-Organizing Genetic Algorithm for Multiple Sequence Alignment

Amouda Nizam[α], Buvaneswari Shanmugham[Ω], Kuppuswami Subburaya[β]

Abstract- Genetic algorithm (GA) used to solve the optimization problem is self-organized and applied to Multiple Sequence Alignment (MSA), an essential process in molecular sequence analysis. This paper presents the first attempt in applying Self-Organizing Genetic Algorithm for MSA. Self-organizing genetic algorithm (SOGA) can be developed with the complete knowledge about the problem and its parameters. In SOGA, values of various parameters are decided based on the problem and fitness value obtained in each generation. The proposed algorithm undergoes a self-organizing crossover operation by selecting an appropriate rate or a point and a self-organizing cyclic mutation for the required number of generations. The advantages of the proposed algorithm are (i) reduce the time requirement for optimizing the parameter values (ii) prevent execution with default values (iii) avoid premature convergence by the cyclic mutation operation. To validate the efficiency, SOGA is applied to MSA, and the resulting alignment is evaluated using the column score (CS). The comparison result shows that the alignment produced by SOGA is better than the widely used tools like Dialign and Multalin. It is also evident that the proposed algorithm can produce optimal or closer-to-optimal alignment compared to tools like ClustalW, Mafft, Dialign and Multalin.

Keywords- Crossover, Genetic Algorithm, Multiple Sequence Alignment, Mutation, Selection, Self-organization.

I. Introduction

Self-organizing system functions without any guidance from the external control (without a central control). Self-organization is done based on local information obtained from the interactions of lower-level components [1]. It is evident from the literature, several GA, a stochastic iterative method [2] are proposed for MSA, to align set of sequences. Major problem of GA, premature convergence can be avoided by blending the concept of self organization and GA. Using SOGA several other problems are solved but applying for MSA with a new mechanism is first of its kind.

About[α]- Centre of Excellence in Bioinformatics, School of Life Sciences, Pondicherry University, Puducherry – 605 014 (corresponding author phone: +91-413-2655212; fax: +91-413-2655211;
E-mail: amouda@yahoo.com
About[Ω]- Centre of Excellence in Bioinformatics, School of Life Sciences, Pondicherry University, Puducherry
E-mail: buvanisuriya@bicpu.edu.in).
About[β]- Department of Computer Science, School of Mathematics and Computer Science, Pondicherry University, Puducherry
E-mail: skswami@yahoo.com

In this case, MSA is defined by the position and gap size in the sequences. Two types of search operators like recombination and gap mutation are included in the algorithm to produce offspring alignments [3]. Apart from these two basic operators, several operators are also proposed in the literature to improve the performance of GA [4-5]. In some case existing GA operators are unsuitable as they are not specific for the problem and the encoded chromosome. This led us to develop new GA operators, specifically for MSA.

The proposed algorithm is illustrated using DNA sequences, but it can be extended to RNA and protein sequences also. A set of *n* DNA sequences of varying length are considered for the alignment process. The nucleotide bases *A*, G, *C*, T corresponds to adenine, guanine, cytosine and thymine and gaps are represented by '-' (hyphen).

The remainder of the paper is organized as follows. The next two section reviews multiple sequence alignment and genetic algorithm. Section 3 explains various methods of the self-organizing genetic algorithm and its advantages over standard GA. Section 4 explain the proposed SOGA with its pseudocode. Section 5 explains the working mechanism of SOGA-MSA with newly developed operators. Section 6 shows the comparison results and discussion. Section 7 is the conclusion and future perspectives.

II. Multiple Sequence Alignment

MSA, aligning three or more nucleotide or amino acid sequences simultaneously is one of the important tasks in bioinformatics. Important application of MSAs is their incorporation in many structure and function prediction methods from sequence. It can reveal conserved residues that enable the identification of possibly important sites. The construction of MSA is closely related to phylogenetic analysis and a phylogenetic tree can be inferred by MSA. The study of molecular evolution is an area where MSA is extensively used [6].

The computation of an optimal alignment mathematically is too complex. Current implementation methods are heuristics in which full optimization is not guaranteed. Various algorithms available for MSA are classified into three main categories: Exact, Progressive and Iterative based on their properties.

Exact algorithms are high quality heuristic in nature, produce very close to optimal alignment. It can handle the only restricted number of sequences and are limited to sums-of-pairs as an objective function.

Progressive alignment using dynamic programming depends on a progressive assembly of the multiple alignments, heuristic in nature but does not guarantee any level of optimization.

Iterative alignment methods produce alignment and refine it through a series of cycles (iterations) until no further improvements can be made. It is deterministic or stochastic depending on the strategy used to improve the alignment. It allows for a good conceptual separation between optimization processes and objective function as its main advantages[7].

The widely used MSA tools implementing different algorithms are ClustalW[8], MultAlin[9], DIALIGN[10], MUSCLE[11], T-Coffee[12], DCA[13]. In addition GA based MSA software like SAGA[14], MSA-GA[3] are available but not in an executable form.

(i) Its flexibility in assigning the fitness function, mathematical function used to evaluate the fitness of the chromosomes.
(ii) The complexity of the MSA process increase exponentially, NP-hard (nondeterministic polynomial) in nature[7] can be solved by GA.
(iii) It is not restricted to need of a particular algorithm to solve the problems. Needs only fitness function to evaluate the chromosomes [15].

III. Genetic Algorithm

GA starts with the generation of population consists of chromosomes, a fixed size encoded solution. Each chromosome represents a possible solution and the space of all feasible solutions is called search space. The role of GA is to alter the generated chromosomes using various operators to get the optimal chromosome with best fitness value in the search space. Iteration continues till the termination condition is satisfied.

Outline of basic GA

1. **[Start]** Generate random population of *n* chromosomes.
2. **[Fitness]** Evaluate fitness *f(x)* of each chromosome *x* in the population.
3. **[New Population]** Create new population using (i to iv) repeatedly until the process is complete
   i) **Selection**
   ii) **Crossover**
   iii) **Mutation**
   iv) **[Accepting]** Place new offspring in a new population.
4. **[Replace]** Use newly generated population for the next iteration.
5. **[Test]** Check the termination condition, if satisfied, stop, and return the best solution.
6. **[Loop]** Go to step 2[16].

IV. Self-Organizing Genetic Algorithm (SOGA)

For a specific input, setting the GA parameters is an important task. The concept of self-organizing GA is to adapt values for parameters like population size, number of generations, selection modes, rates of selection crossover and mutation during execution.

In the blend of SO and GA, most of the parameters change according to the fitness of the chromosomes. An attempt towards SOGA requires a complete understanding of the relationship among various parameters and its impact in the performance.

The aim of SOGA is to create an automated computer program that solves the problem with little or no information from the user. The difficulty in choosing the appropriate number of generations, chromosome length, crossover and mutation rate is eliminated, thus GA is made efficient and simple to use.

Using GA, solutions to a particular problem are not algebraically calculated rather found by a population of solution alternatives, which are altered (using operators like crossover and mutation) in iterations of the algorithm in order to increase the probability of having better solutions. In optimization, better chromosomes with higher fitness value will be selected.

SOGA over Standard Genetic Algorithm (SGA)

Encoding of chromosomes in SGA is usually fixed-length strings. In SOGA, the length of the chromosome can be made to change adaptively based on the problem[17].

Population Size is fixed in SGA and the corresponding number of chromosomes is generated. Population size 50-100 is reported as best[18]. Population size can be made to change adaptively based on the problem.

- It can be self- organized by generating both small and large populations and the fitness value of each of the chromosomes is calculated. If the average fitness of the larger population is higher than the smaller population then the program continues with the larger population otherwise with the smaller population.
- Each time at convergence, population size is doubled till it reaches an upper limit[19].

Number of Generations is always fixed in SGA, and the algorithm terminates on reaching specified number of generations or fitness level or at convergence. An optimal solution may not be reached if termination is due to the maximum number of generations. Hence it

is necessary to self-organize number of generations based on the problem.

Selection Operator in SGA is usually one or combination of operators. In SOGA, certain conditions are defined to choose the appropriate operator for a particular problem for e.g. based on the average fitness of the generated chromosome.

Crossover/ Mutation Operator in SGA is usually one or combination of operators, and it can be self-organized

- By defining conditions based on which the appropriate operator or rate is chosen.
- Crossover/ Mutation operation is performed with a specified number of methods and based on the average fitness of the resulting chromosome, an appropriate method is chosen[20].
- The algorithm can be executed initially with a minimum optimal crossover/ mutation rate. At each point of convergence, instead of termination the rate can be increased cyclically till it reaches the optimal upper limit[21].
- The crossover/ mutation rates adapted from high to minimum optimal rate [22].
- Along with the chromosome generated with the current value obtained by increase or decrease in the rate, chromosomes corresponding to larger and smaller are also generated. The chromosome with higher fitness is chosen[23] as an elite.

It is reported in the literature that generally crossover rates should be high (80%-90%) and mutation rate should be very low (0.5%-1%)[18].

Advantage of SOGA

- GA with self-organizing coding, operators and parameter values is efficient and simple to use.
- Time required for optimizing parameter values is eliminated by using SOGA. In SGA, optimal parameter value can be found by executing with all possible values and combinations with other parameters.
- The default parameter values assumed to be optimal is considered when the user fails to select appropriate values. Even this value may lead to bad results for some problems. Instead of getting parameter values from user SOGA self-organizes all or most of the parameters to assign values based on the problem.

V. Proposed SOGA

SOGA proposes two new operators to perform crossover and mutation. The proposed Self-Organizing Crossover Operator **(SOCO Operator)** selects an appropriate crossover point and the corresponding rate from the initial crossover point. The proposed mutation operator **(Self-Organizing Binary Shuffler)** converts the chromosome representation into a binary form and performs mutation for a range of rates till the termination condition is satisfied. The number of generations is also self-organized, which varies depending on the problem.

Pseudo code of the SOGA

1. **[Start]** Generate the random population of *n* chromosomes.
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population.
3. **[Selection]** Select and save the elite (chromosome with highest fitness value) in the current population.
4. **[New Population]** Create a new population using (i to iv) repeatedly until the process is complete

i) **[SOCO]** Self-organizing the selection of crossover point based on the specified optimal rate and perform crossover using **SOCO operator**.

ii) **[Selection]** Select and save the elite in the current population.

iii) **[SOBS]** Convert the chromosome representation to a binary form and perform mutation for a range of rates cyclically using **SO Binary Shuffler**.

iv) **[Selection]** Select and save the elite in the current population.

5. **[Test]** Check the termination condition. If satisfied, stop, and return the best solution.
6. **[Loop]** Go to step 4.

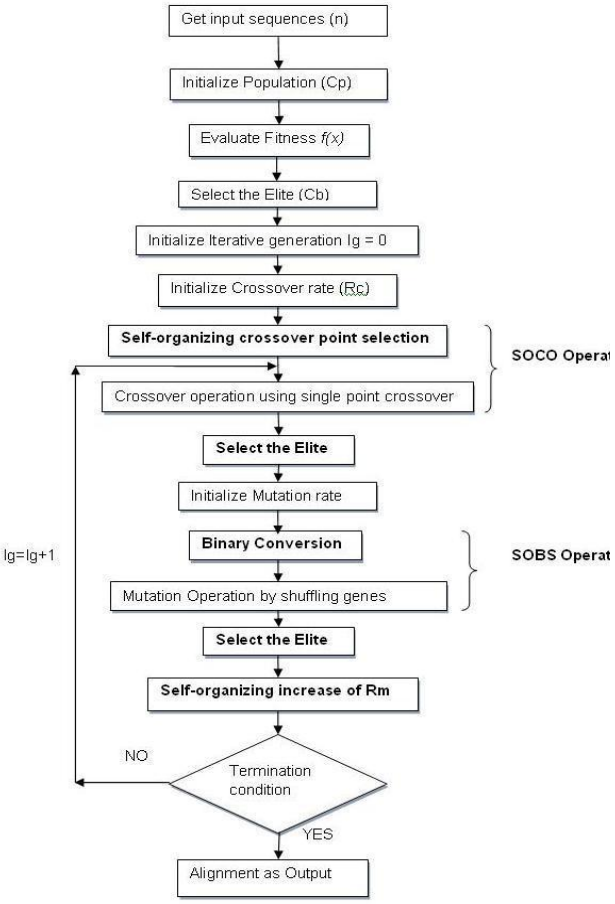

Fig.1. Flow chart of SOGA.

VI. SOGA-MSA

a) *Chromosome Representation*

In general, chromosome is a matrix with fixed lengths and represented as sequences with spaces[25,26]. For the problem of MSA, the gap positions are used to encode the chromosome. The number of gaps to be inserted in each sequence is calculated in such a way that the length of all sequences in the alignment (global) is same. A single chromosome consists of gap positions of all the sequences in order. In the mutation process, the chromosomes are encoded as binary digits (1, 0) representing presence and absence of the gap in sequence. In SOGA, the length of the chromosome is adaptively changed based on the number of sequences and its length[17].

b) *Number of Generations*

In each generation, the algorithm generates chromosomes of required population, and its fitness score is evaluated. Chromosomes from the current population are stochastically selected and modified by crossover and mutation, which undergo next generation. As the rate of mutation is made to increase cyclically based on the fitness value, iteration completes only when the optimal upper limit is reached. The number of generations depends on the betterment of the fitness value obtained in each generation.

For e.g., consider the dataset 469 with three sequences as input. The generation starts with Rm=1% produces an alignment with CS = 36. Next generation continues with Rm 1% resulting further no increase in CS. Hence by the concept of self-organization Rm increased to 3% resulting CS = 37. Self-organizing process continues till the upper limit of Rm (80%) is reached. In 43 generations the CS of the output alignment is 45 as shown in the table I.

Table 1: Example For Self-Organizing Number Of Generations

Iterative Generation (Ig)	Mutation rate (Rm)	Column Score (CS)
1	1%	36
3	3%	37
21	37%	40
25	43%	44
41	79%	45
43	81%	45

c) *Population Initialization*

Population size indicates the number of chromosomes in a generation, and it must be optimal for a particular problem.

Table 2: Example For Population Initialization

Sequence	Sequence Length	No. of Gaps	Gap positions	Sorted gap positions	Alignment
TCTAGATG	8	6	5|0|11|3|6|9|	0|3|5|6|9|11|	-TC-T--AG-A-TG
CTATGATGTA	10	4	12|10|0|7|	0|7|10|12|	-CTATGA-TG-T-A
ACGATGTA	8	6	7|4|11|5|8|13|	4|5|7|8|11|13|	ACGA--T--GT-A-
GTTCTAT	7	7	8|4|6|1|13|3|0|	0|1|3|4|6|8|12|	--G--T-T-CTA-T
ACGTATAGCAAT	12	2	9|4|	4|9|	ACGT-ATAG-CAAT

Best population size also depends on encoding method and size of the encoded string. According to research, it is proven that increase in population size after a limit does not improve the performance of GA[18].

Considering m sequences to be aligned with the length ($m_1,\ldots m_i$) and the space ratio r_{sp} =0.2. If the longest length of sequences to be aligned is m_{max}, then N = m_{max} * (1+ r_{sp}). The value of N is the size of search space of alignments. It limits the longest length of alignments that chromosomes can represent.

Chromosomes can be transformed to actual alignments by inserting gaps in the appropriate positions. For e.g., m_{max} = 12, r_{sp} = 0.2, then N = 14.

d) *Fitness Evaluation*

The fitness function returns a numerical score indicating fitness of the candidate alignment. It is an important parameter to determine which alignment will survive in the next generation. The fitness is evaluated by calculating the (CS) column score. CS = EM/ AL, where AL is the alignment length, EM (Exact match) =1, when all the base pair in the entire column is aligned with the same base pair.

e) *Selection*

SOGA implements an elitism operator, where an elite is the chromosome with best fitness value. The process comprises the following processes

(i) Evaluate the column score.
(ii) Sort the chromosomes.
(iii) Select and save the elite.

With the current population, SOGA undergoes a crossover. New chromosomes are generated and the elite is selected. If the fitness value of new one is greater, elite is replaced else the process continues with the same. In the same way for mutation elite selection and comparison process is repeated. This process continues for every generation to ensure that the elite saved at the end is best.

A new mechanism is followed for crossover and mutation operation in self organizing GA.

f) *Self-Organizing Crossover Operator (SOCO)*

In single point crossover operation [22], the crossover point is selected initially for a particular rate. Then the genes from starting point to the crossover point are copied from one chromosome and the rest from the second chromosome.

In SGA, crossover for a particular rate may lead to the occurrence of crossover point within a sequence itself. It may create problems like

(i) Increase in the number of gaps for a particular sequence.
(ii) Occurrence of repeated gap positions in a sequence.

To overcome this major disadvantage, proposed operator SOCO defines a new point called complete point. Each complete point refers to the end position of each sequence in a chromosome. The number of complete point in a chromosome is based on the number of input sequences. For e.g. if input sequences are five, then the chromosome contains four complete points as shown in Fig.2.

The new working principle followed by SOCO operator is as follows:

(i) Initialize the crossover rate (Rc) and select the corresponding crossover point.
(ii) Selects the complete point near the default crossover point.
(iii) Performs single point crossover operation.
(iv) Generates MSA corresponding to the chromosome.
(v) Calculates fitness score.
(vi) Selects elite.

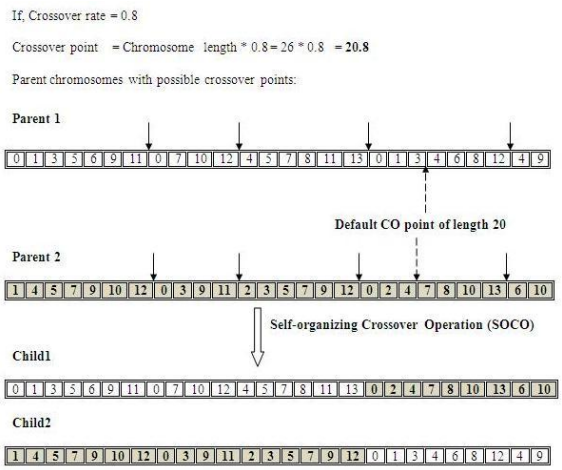

Fig.2. Example for Self-Organizing Crossover.

g) *Self-Organizing Binary Shuffler (SOBS)*

In SGA, either an optimal mutation rate which is unsuitable for all inputs is fixed or selected from a range of rates given as optional. It is hard for the user to select appropriate rate without the knowledge of the problem. To eliminate these problems, a new mutation operator with a different approach is proposed. Instead of a fixed rate, the operator performs mutation for a range of rates cyclically [21, 24] till the termination condition are satisfied.

In default shuffling process for mutation leads to the problem like

(i) Increase in the number of gaps for a particular sequence.
(ii) Occurrence of repeated gap positions in a sequence.

To avoid this, proposed mutation operator involves conversion of chromosome representation to binary digits (1,0) represents the presence and absence of gaps. The new working principle followed by SOBS operator is as follows:

i) Converts the chromosome representation to a binary form.
ii) Initialize minimum optimal mutation rate

and the corresponding mutation point is selected.

iii) Genes before mutation point are considered for mutation.

iv) The genes within each complete point and if any gene occurs between the last complete point and mutation point are shuffled separately as shown in Fig. 3.

v) Change chromosome representation to gap positions.

vi) Generates MSA corresponding to the chromosome.

vii) Calculates fitness score.

viii) Selects elite.

If the elite is replaced by the selection condition, the generation continues with the same rate else increases cyclically until an optimal upper limit is reached. The algorithm terminates on reaching the optimal upper limit when no further increase in the column scores.

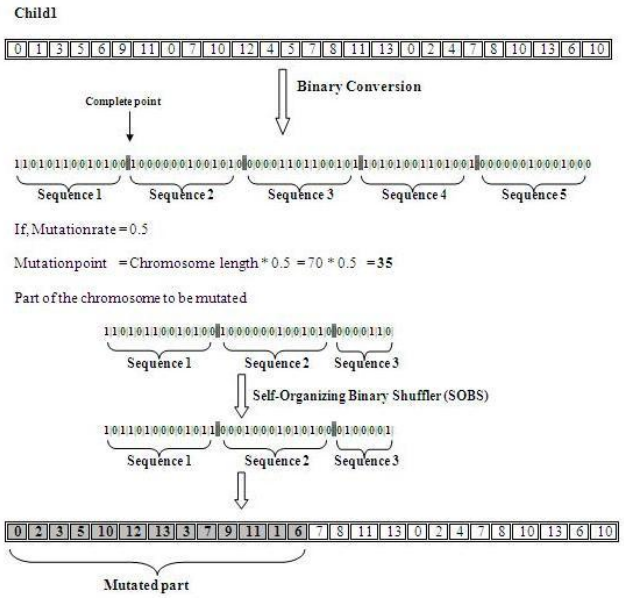

Fig.3. Example for Self-Organizing Mutation

VII. Results and Discussion

To validate the proposed algorithm, various parameters and results of SOGA is compared with SGA. The dataset 469 from oxbench_mdsa_all with three sequences is used as input. The comparative results show that on average, SOGA-MSA produces better results than the SGA-MSA. As an advantage, in fewer numbers of generations and time SOGA-MSA produces the better alignments as tabulated below.

Table 3: Comparison of Sga and Soga on Msa

GA	Population size	Crossover rate (Rc)	Mutation rate (Rm)	No. of Generations	Exact match (EM)	Alignment Length (AL)	Column Score (CS)
SGA-MSA	100	80	60	50	44	406	0.10
SOGA-MSA	100	70	1-80	43	45	406	0.11

The alignments produced by the widely used MSA tools with default parameter settings are compared with the developed **SOGA-MSA,** and the results are tabulated. The standard reference datasets of DNA sequence alignments from BAliBASE[27] are used as input.

The results of two dataset given below are tabulated.

- Dataset RV11_BBS11022 from the mdsa_all version with four sequences.
- Dataset RV11_BBS11002 from the mdsa_100 version with eight sequences

Table 4: Comparison of Performance of Soga-Msa and Other Msa Tools

DATASET **RV11_BBS11022**				DATASET **RV11_BBS11002**			
MSA Tool	Exact Match	Alignment Length	Column Score (CS)	MSA Tool	Exact Match	Alignment Length	Column Score (CS)
Dialign	6	274	0.021	ClustalW	1	232	0.004
Mafft	11	208	0.052	Multalin	0	220	0
SOGA-MSA	**9**	**248**	**0.036**	**SOGA-MSA**	**1**	**259**	**0.003**

The results produced by SOGA-MSA and other tools like Dialign, Mafft, ClustalW and Multalin are tabulated above. It is observed that the CS of the alignment produced by SOGA-MSA is better than most commonly used tools like Dialign, Multalin and equal to ClustalW. The betterment of the multiple sequence alignment compliments the efficiency of the proposed algorithm.

VIII. Conclusion and Future Perspectives

In SOGA-MSA parameters like number of generations, chromosome length, crossover and mutation rate are made to adapt the values during execution, whereas in standard GA these values are determined before execution. In general, the values of various parameters of GA based algorithm are either default or selected from options. It is hard for a non-specialist to assign the values of various parameters without complete knowledge of the problem. Even default values may lead to bad results for some input. This is completely facilitated and proven by the proposed self-organizing approach of GA for MSA, where the parameter values are chosen by itself. The main advantage of SOGA-MSA is getting sequences alone as input from the user. Premature convergence considered as one of the major fitness range problems of standard GA is completely avoided by the execution for a range of rates.

The self-organizing crossover and mutation operator developed for MSA prevents the problem of repetition and increase of gaps in chromosomes. In addition, elitism selection avoids disruption of the best chromosome. The proposed SOBS self-organize the increase in mutation rate, which explores all rates within the range. As an advantage, this mechanism ensures that the best alignments produce for varying rates within the range are also included in the process of alignment.

Several widely used MSA tools like DCA[13] has a strong limitation in the number of sequences it can handle. In SOGA-MSA, there is no limitation in the number of input sequence and its length.

The algorithms used in other tools will produce the alignment with the same column score for every execution. However, in SOGA-MSA implementing the stochastic iterative algorithm, there is a chance of generating better alignments than the previous alignment in each execution. Further, it may generate better alignments with an increase in the number of generations also.

The future objectives are to self-organize other parameters like population size, crossover rate, etc., to minimize the execution time and to improve the quality of the alignment further.

References Références Referencias

1. T. D. Seeley, "When Is Self-Organization Used in Biological Systems?," *Biol. Bull.,* 2002, 202(3): 314–318.
2. J. H. Holland, "Adaptation in natural and artificial systems", University of Michigan Press, Ann Arbour, MI, 1975.
3. C. Gondro, B. P. Kinghorn, "A simple Genetic Algorithm for multiple sequence alignment", *Genet. Mol. Res.,* 2007, 6 (4): 964-982.
4. N. Kubota, T. Fukuda, K. Shimojima, "Virus-evolutionary genetic algorithm for a self-organizing manufacturing system", *Computers and Industrial Engineering,* 1996, 30(4): 1015-1026.
5. S. S. Ray, S. Bandyopadhyay, S. K. Pal, "New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering", Springer-Verlag Berlin Heidelberg, 2005, pp. 605–610.
6. S. Diamantis,C. Anna, "Comparison of Multiple Sequence Alignment programs", M.Sc. Bioinformatics, National and Kapodistrian University of Athens.
7. C. Notredame, "Recent progresses in multiple sequence alignment: a survey", *Pharmacogenomics,* 2002, 3(1): 131-144.
8. J. D. Thompson, D. G. Higgins, T. J. Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting position specific gap penalties and weight matrix choice", *Nucleic Acids Res.,* 1994, 22: 4673-4680. http://www.ebi.ac.uk/Tools/clustalw/
9. F. Corpet, "Multiple sequence alignment with hierarchical clustering", Nucleic Acids Res., 1988, 16: 10881-10890. http://bioinfo.genotoul.fr/multalin/multalin.html
10. B. Morgenstern, A. Dress, T. Wener, "Multiple DNA and protein sequence based on segment-to-segment comparison", *Proc. Natl. Acad. Sci.*, 1996, 93: 12098-12103. http://bibiserv.techfak.uni-bielefeld.de/dialign/
11. R. C. Edgar, "MUSCLE: multiple sequence alignment with high accuracy and high throughput", *Nucleic Acids Res.,* 2004, 32: 1792-1797. http://www.ebi.ac.uk/Tools/muscle/
12. C. Notredame, D. G. Higgins, J. Heringa, "T-Coffee: A novel method for fast and accurate multiple sequence alignment", *J Mol Biol.,* 2000, 302: 205-217. http://www.ebi.ac.uk/Tools/t-coffee/
13. J. Stoye, V. Moulton,A. W. Dress, "DCA: an efficient implementation of the divide-and conquer approach to simultaneous multiple sequence alignment", *Comput. Appl. Biosci.,* 1997, 13(6): 625-626. http://bibiserv.techfak.uni-bielefeld.de/dca/
14. C. Notredame, D. G. Higgins, "SAGA: sequence alignment by Genetic algorithm", *Nucleic Acids Res.,* 1996, 24(8):1515-1524.
15. K. Karadimitriou, D. H. Kraft, "Genetic Algorithms and the Multiple Sequence

Alignment Problem in Biology", In Proc. 2nd Annual Molecular Biology and Biotechnology Conference, Baton Rouge, LA, 1996.
16. S. A. Fatumo, I. O. Akinyemi, E. F. Adebiyi, "Aligning Multiple Sequence with Genetic algorithm", *International Journal of Computer Theory and Engineering,* 2009, 1(2): 179-182.
17. S. Wu, M. Lee, T. M. Gatton, "Multiple Sequence Alignment using GA and NN", International Journal of Signal Processing, *Image Processing and Pattern Recognition*, 21-30.
18. Introduction to genetic algorithm tutorial in www.obitko.com/tutorials/genetic-algorithms (Last accessed: 7.12.2010).
19. G. R. Harik, F. G. Lobo, "A Parameter-Less Genetic Algorithm", *IEEE Transactions on Evolutionary Computation*, 1999: 523-528.
20. T. Hong,H. Wang, W. Lin, W. Lee, "Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process", *Applied Intelligence*, 2002, 16: 7–17.
21. H. Bao-Juan, Z. Jian, Y. De-Hong, " A Novel and Accelerated Genetic algorithm", *WSEAS Transactions on Systems and Control,* 2008, 3(4): 269-278.
22. R. Breukelaar, T. Bäck, "Self-Adaptive Mutation Rates in Genetic Algorithm for Inverse Design of Cellular Automata", July 2008: 12–16.
23. D. Thierens, "Adaptive mutation rate control schemes in genetic algorithms", Institute of Information and Computing Sciences, Utrecht Univerisity, The Netherlands, 2002.
24. J. Zhang, J. Zhuang, H. Du, S. Wang, "Self-organizing genetic algorithm based tuning of PID controllers", *Information Sciences*, 2009, 179 (7): 1007-1018.
25. J. T. Horng, E. M. Lin, B. H. Yang, E. Y. Kao, "A Genetic Algorithm for multiple sequence alignment", In Proc. of the GCB, 2001.
26. D. Liu, X. Xiong, Z. Hou, B. D. Gupta, "Identification of motifs with insertions and deletions in protein sequences using self-organizing neural networks", *Neural Networks,* 2005, 18 (5-6): 835-842.
27. H. Carroll, W. Beckstead, T. O'Connor, M. Ebbert, M. Clement, Q. Snell, D. McClellan, "DNA reference alignment benchmarks based on tertiary structure of encoded proteins", *Bioinformatics,* 2007, 23(19): 2648– 2649. http://dna.cs.byu.edu/mdsas/download.shtml