Self-Organizing Genetic Algorithm: A Survey

Amouda Nizam
Centre for Bioinformatics,
School of Life Sciences,
Pondicherry University
Puducherry, INDIA

Buvaneswari Shanmugham
Centre for Bioinformatics,
School of Life Sciences,
Pondicherry University
Puducherry, INDIA

ABSTRACT
Self-organization systems are an increasingly attractive dynamic processes without a central control, emerge global order from local interactions in a bottom up approach. The advantage of blending the concept of self-organization enhances the working efficiency of other techniques to find a solution of huge search problem. Genetic Algorithms (GA) is such a technique, inspired by the natural evolution process, used to solve difficult optimization problem of large space solution, for an example, multiple sequence alignment (MSA) problem in a bioinformatics research. Self-organization technique automates the selection of appropriate parameter values of GA during execution without the user's intervention. An attempt towards applying Self-organizing Genetic Algorithm (SOGA) on MSA requires a complete knowledge of the various parameters of SO and its relationships. This lead us to make a complete survey on inherent properties of SO and the method of blending GA in order to develop a self-organizing genetic algorithm (SOGA) for MSA. The aim of the research is to make use of the efficiency of GA without getting any input from the non-trained users to tune the parameters in order to achieve the expected result.

General Terms
Genetic Algorithm, Multiple Sequence Alignment, Self-organization

Keywords
Crossover, Mutation, Selection, Self-organizing genetic algorithm

1. INTRODUCTION
Self-organizing system are a physical, chemical or biological system that takes a form that is not imposed by an external directing influence *i.e.,* without a central control. Self-organizing systems are designed as sets of similar lower-level components, interacting conceptually and physically in order to obtain the pattern at global level of a system. The principal challenge is to understand how the lower-level components interact to produce a common pattern. These components may interact directly or indirectly, depending on the influence of behavior of one component affecting the behavior of the other components [1-3].

GA is used to solve difficult search/ optimization problems and machine-learning problems that have previously resisted automated solutions quickly and reliably. These algorithms are easy to interface with existing simulations and models, and they are easy to hybridize [4]. The GA solutions, for a particular problem, are not algebraically calculated, rather found by a population of solution alternatives which is altered (using operators like crossover and mutation) in each iterative step of the algorithm in order to increase the probability of having better solutions. In optimization, the solutions to a particular problem will be selected accordingly how well they solve the problem, is denoted by its fitness value. GA explores the multi-parameter space of solution alternatives for a particular problem. In each iterative step, chromosomes are altered, leading the population to even more promising regions of the search space [5]. To blend self-organization and GA, it is essential to understand its various aspects clearly.

For Genetic Algorithm (GA), several operators and encoding methods are proposed in the literature [6-13]. Different operators are selected depending on the problem and also the method of encoding the chromosomes. Complete knowledge about the problem is required to select the appropriate encoding method, operators and parameter values. For non-trained users, the selection of appropriate parameters is difficult. These difficulties can be solved by self-organizing the GA for a particular problem by which the processing system is converted into a self-organizing system, which solves the problem without getting any input from the users.

The remainder of the paper is organized as follows. The next section describes about the self-organizing systems. Section 3 explains the components of GA and different ways to self-organize GA. Section 4 explains mapping of self-organization and GA. Section 5 presents a brief comparison between the existing Multiple Sequence Alignment (MSA) methods and GA. Section 6 explains Self-organizing Genetic Algorithm (SOGA) for MSA. Section 7 briefly discusses SOGA variants available and their parameter settings. Shortcomings of SOGA are stated in section 8.

2. SELF-ORGANIZING SYSTEMS
Many natural systems become structured by their own internal processes. Pattern at the global level of a system emerges solely from numerous interactions among the lower-level components of the system *i.e.,* interactions internal to the system, without intervention by external direct influences. Components achieve a simple task individually, but a complex collective behavior emerges from their mutual interactions. Such a system modifies its structure and functions to adapt changes based on previous experience [1].

Self-organization is an algorithmic approach adopted by the system that has the ability to adjust its own structure by local interactions without external interventions that emerges global behavior *i.e.,* self-control system, not environment dependent works in a bottom-up approach. Whereas self-adaptive systems are architecture oriented, centralized system adopts the mechanism, to reconfigure its global behavior and looks for other possibilities when it is not achieving the desired result. *i.e.,* decentralized control system, environment

dependent works in a top-down approach. The most current techniques used for designing self-organizing applications are direct translations of natural self-organizing mechanisms like magnetism, crystallization, pigmentation patterns on shells etc.

The two basic modes of interaction [1] among the components of self-organizing systems are:

Positive feedback: A process in which synthesis of a substance over a thresh-old level produces response stimulating its synthesis. As an example, the hormone oxytocin stimulates muscular contractions of the uterus, which in turn stimulates the release of more oxytocin [14].

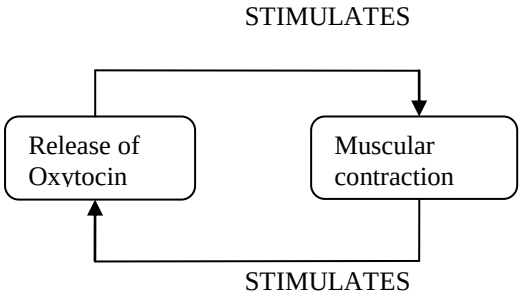

Fig 1: Example for positive feedback

Negative feedback: A process in which synthesis of a substance over a thresh-old level produces response inhibiting the synthesis of the substance. As an example, a rise in blood sugar leads to the production of insulin. As insulin level rises, glucose is removed from the blood [14].

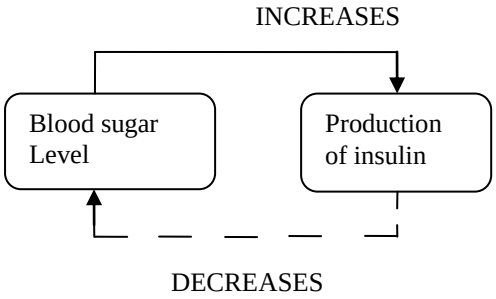

Fig 2: Example for negative feedback

3. SELF-ORGANIZING GENETIC ALGORITHM

The aim of self-organizing genetic algorithm is to create an automated computer program that solves the problem with little or no information from the user. Hence the number of external parameters is reduced.

3.1 Parameters of Genetic Algorithm

Setting the parameters of GAs is a non-trivial task. GA is self-organized by adapting values for parameters like population size, number of generations, modes of selection, crossover and mutation or the selection rate, crossover probability and mutation probability during the execution process. In SOGA, most of the GA parameters change according to the fitness of chromosomes.

3.1.1 Encoding a Chromosome

A chromosome contains information which represents the solution. Each gene in the chromosome can represent some characteristics of the solution or the whole string can represent a number. Selection of encoding methods depends mainly on the problem. Various encoding methods are binary, permutation, tree and value encoding [15].

3.1.2 Population Size

Population size indicates the number of chromosomes in a population for a single generation. By increasing the population size, GA can get better chromosomes which reduce the error in decision-making. By decreasing the size of population, the GA can converge faster. The population size is a critical parameter in a GA. Best population size depends on the problem, length of the chromosome and encoding method. If there are too few chromosomes, GA has a few possibilities to perform crossover and only a small part of search space is explored and the GA will converge to sub-optimal solutions. If there are too many chromosomes, GA slows down. It was shown [15] that increasing population size after some limit does not improve the performance of GA.

3.1.3 Number of Generations

In each generation, GA involves generation of chromosomes of required population. The fitness of each chromosome is evaluated. From the current population, chromosomes are stochastically selected and modified (cross over and mutation) to get a new population.

3.1.4 Operators of Genetic Algorithm

The commonly used operators of GA are:

3.1.4.1 Selection Operator

This operator performs the equivalent role to natural selection. Use of a selection operator is to choose the fit chromosomes from the population of the current generation to the next generation. Selection rate should not be very low or very high. At very low selection pressures *i.e.*, rate, GA is not able to discriminate between the fit and the unfit chromosomes. At very high selection pressures, GA only pays attention to the best chromosomes, resulting in an immediate loss of diversity and little recombination to be done. Different selection methods reported in literature are best, elitist, fitness-proportionate, generational, hierarchical, random, rank, Roulette-wheel, scaling, top-percent, tournament and truncation selection [16, 17].

3.1.4.2 Crossover Operator

Crossover is a genetic operator that combines two chromosomes in order to produce a new chromosome possessing some characteristics of each parent. The key idea of crossover is that the child may be better than both of the parents if it takes the best characteristics from each of the parents. Crossover methods available are single point, double point, uniform, arithmetic [15] and heuristic crossover [16].

3.1.4.3 Mutation Operator

Mutation operator alters one or more gene values in a chromosome from its initial state resulting in entirely new chromosomes, helps in arriving at a better solution. Mutation is considered to be important as it helps to prevent the population from stagnating at any local optimum. Mutation occurs according to a certain mutation probability. The probability should generally be set to a low value. If it is too high, the search will turn into a primitive random search. The

mutation operator enhances the ability of the GA to find a near optimal solution to a given problem by maintaining a sufficient level of genetic variety in the population, which is needed to make sure that the entire solution space is used in the search for the best solution [18]. Mutation methods available are flip bit, boundary, non-uniform, uniform and Gaussian mutation [16].

3.2 Methods to Self-organize GA

3.2. 1 Self-organizing Encoding

In GA, the chromosomes in the population are usually encoded as fixed-length strings. In SOGA, the length of the chromosome can be made to change adaptively based on the problem [6].

3.2.2 Self-organizing Population Size

Population size can be made to change adaptively based on the problem.

- Population size can be self-organized by generating both small and large populations. Fitness of each of the chromosome is calculated. If the average fitness of the larger population is higher than the smaller population then the program continues with the larger population, if not with the smaller population.
- Each time at convergence, the population size is doubled till it reaches an upper limit [19].

3.2.3 Self-organizing number of Generations

In GA, the algorithm terminates when a condition of a specified number of generations or a satisfactory fitness level has been reached or convergence (no further increase in fitness score). If the termination of the algorithm is due to a maximum number of generations, an optimum solution may not have been reached. Hence it is necessary to self-organize the number of generations based on the problem.

3.2.4 Self-organizing Selection Operator

In GA, the choice of selection operator is usually one or combination of more than one operator. In SOGA, certain conditions are defined to choose the appropriate selection operator for a particular problem for example based on the average fitness of the generated chromosome.

3.2.5 Self-organizing Crossover/ Mutation Operator

- The choice of the operator/ rate can be self-organized by defining conditions based on which appropriate operator/ 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 [7].
- The algorithm can be executed initially with a minimum optimal crossover/ mutation rate. At each point of convergence, the rate can be increased cyclically, till it reaches the optimal upper limit [8, 11].

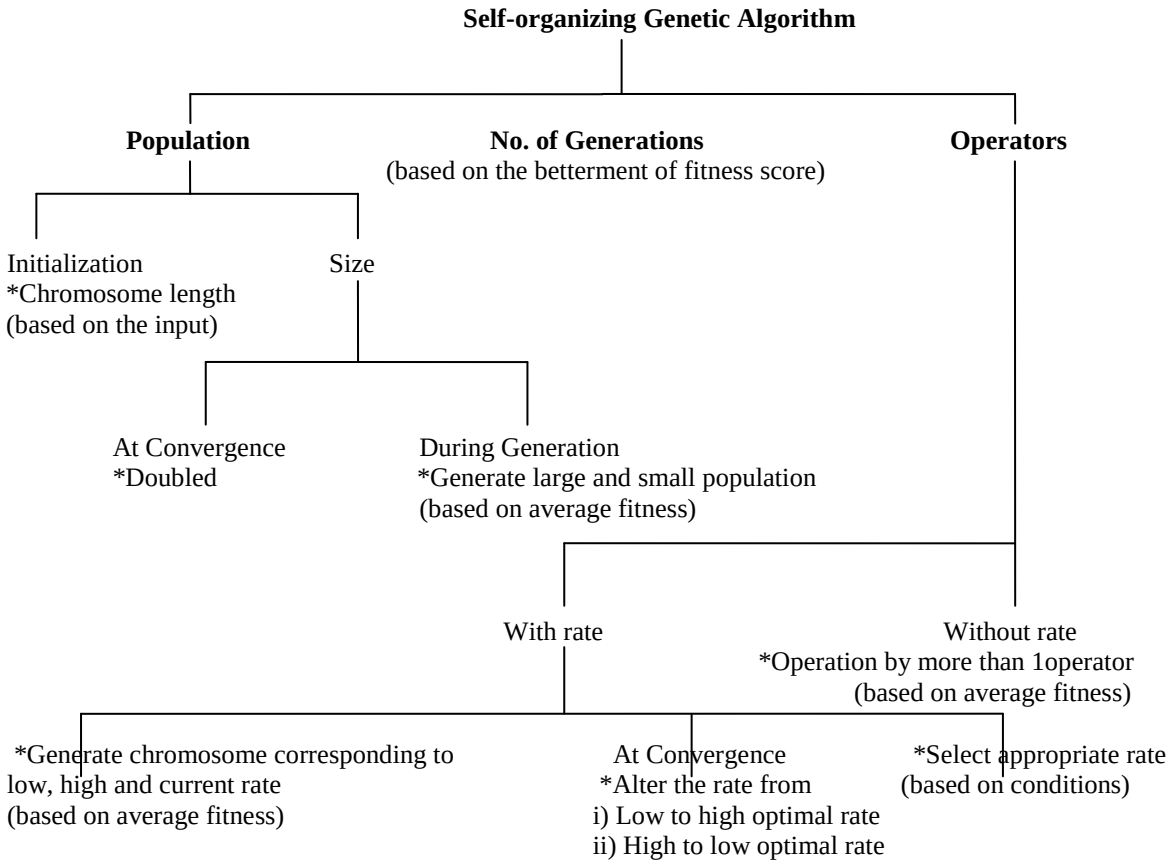

Fig 3: Self-organizing Genetic Algorithm

- The crossover/ mutation rates adapted from an initial very high rate to a minimum optimal rate [9].
- Chromosomes corresponding to the larger and smaller crossover/ mutation rate are generated. In addition generate a chromosome corresponding to the current value obtained by increase or decrease in the rate. The chromosome with higher fitness is chosen [10]. GA can also be self-organized by implementing a conditional increase in the rate of both crossover and mutation until its corresponding optimal upper limit is reached.

Apart from the three basic operators discussed above, there are some low-level operators like dominance, inversion, recording, deletion, segregation, diploidy, translocation, duplication, sexual differentiation and higher-level operators like niche and speciation [20].

4. PROPERTIES OF SELF-ORGANIZATION MAPPED TO GENETIC ALGORITHM

The correlation or coherence between separate components produced by self-organization defines an ordered configuration. Organization can be defined as the characteristic of being ordered or structured so as to fulfill a particular function. The individual properties of self-organization are discussed below:

Systemness and complexity
Self-organization takes place in a system, in a coherent whole that has parts, interactions, structural relationships, behavior, state, and a border that delimits it from its environment. Self-organization systems are complex systems and its complexity depends on the number of elements and connections among them, the system's structure. SOGA applied to a particular problem is a complex system generating optimal solution by various operations.

Control Parameters
Self-organizing system consist of a set of parameters influencing the state and behavior of the system. Control parameters of SOGA are chromosome length, fitness value, crossover/mutation rate and its range, population size and betterment of the results.

Information Production
Self-organizing systems are information-producing systems. During execution of SOGA various information like chromosomes, its fitness value, and optimal crossover/ mutation rate are produced by various interactions of these components.

Dynamic and Global Order from Local Interactions
Self-organizing systems are characterized by multiplicity of interactions. This emphasizes that the SO systems are dynamic and require continual interactions among the lower - level components for the emergence of global organization. In the self-organized state all segments of the system are strongly correlated. In SOGA value of a particular parameter is decided by a dynamic interaction among various other parameters, for an example self-organizing selection of suitable crossover point depends on the chromosome length, initial rate and number of in-put sequences. The process

continues for a number of generations until the termination condition is satisfied.

Decentralized Control
Decentralized control refers to a particular "architecture of information flow" in the absence of external control. Individuals in self-organized social groups do not rely on instructions from well informed individuals. In SO systems, control of the organization is typically distributed over the whole system. All parts contribute evenly to the resulting arrangement. When SOGA is applied to a problem, almost all or most of the parameters values are assigned in a self-organizing manner without a centralized control.

Critical Values, Fluctuation and Intensification
In SO systems, if certain critical values of the control parameters are reached, structural change takes place and the system enters a phase of instability and criticality. Even small disturbances from inside the system intensify themselves and initiate the formation of order. For *e.g.* in SOGA crossover/ mutation rate can be made to increase cyclically when there is no betterment of the fitness value. On reaching the upper limit of the rate the process terminates. These are critical values deciding the number of generation and hence termination of GA.

Robustness and Resilience
Self-organizing systems are relatively insensitive to perturbations or errors, and have a strong capacity to restore by themselves. One reason for the fault-tolerance is redundant and distributed organization: the non-damaged regions can usually make up the damaged ones. Outside a critical phase, the structure of the system is relatively stable concerning local disturbances and a change of boundary conditions. Another reason for the intrinsic robustness is that self-organization thrives on randomness, fluctuations or "noise". SOGA randomly generates a population of solutions which are modified by various operators. From this initial and population modified in various generations, the best or top scoring chromosomes will be selected. Saving best solution after each process ensures that the solution at the end is best of all generation.

Stigmergy and Dense Heterarchy
It refers to the mechanism by which members of the group coordinate their activity based on previous behavior. The stimulation of the members by the very performances they have achieved is a significant one, inducing accurate and adaptable responses. Self-organizing system shows a stigmergic information flow. In addition to stigmergic information flow, direct communication between individuals or between groups and individuals is also important. Dense heterarchy indicates the communication of each individual with any other. In SOGA with self-organizing mutation, rate will be increased cyclically when fitness value converge. In this process, increase in fitness rate depends on the convergence of fitness value in previous generation.

Openness
Self-organization can only take place if the system imports entropy that is transformed; as a result, energy is exported or dissipated. SOGA is an open system that accepts the changes in previous steps which modifies accordingly and carries it to next steps.

Non-linearity and Feedback and Circular Casualty
In SO systems, the relation between cause and effect is much less straightforward: small causes can have large effects, and large causes can also result in small effects. In a critical phase, causes and effects cannot be mapped linearly; similar causes can have different effects and different result in the same effect. Feedback loops (Fig.1 and 2) occur within a self-organization system; circular casualty [1] involves a number of processes like $p1, p2 \ldots pn\ (n >= 1)$, and $p1$ results in $p2$, $p2$ in $p3 \ldots pn-1$ in pn and pn in $p1$. for e.g. the fitness value is critical in assigning the crossover/ mutation rates between generations. Even a small increase in fitness value can help in retaining the same rate. In GA chromosomes from initial population undergo crossover, resulting population undergo mutation and the cycle continues till termination.

Bifurcation and Selection
Bifurcation means a phase transition from stability to instability or a sudden transition from one pattern to another following even a small change in the parameter of the system. Small adjustments in such parameters can induce large changes in the state of the system. Once a fluctuation intensifies itself, the system enters a critical phase where its development is relatively open, certain possible paths of development emerge, and the system has to make a choice. This means dialectic of necessity and chance. In a critical phase that can also be called the point of bifurcation, a selection is made between one of several alternative paths of development. In SOGA with a self-organizing population size, fitness value convergence act as the point of bifurcation. At convergence, population size is increased.

Inner Conditionality
Self-organizing systems are influenced by their inner conditions and boundary conditions from their environment. The parameters of SOGA are assigned based on various conditions and interactions of the components.

Relative Chance
In SO systems there is dialectic of chance and necessity; certain aspects are determined whereas, others are relatively open and subject to chance. In some cases SOGA parameters like initial population size, selection/ crossover/ mutation rate are determined whereas, chromosome length, optimal rates and number of generations are assigned in a self-organizing manner during execution.

Symmetry Breaking, Organizational Closure and Emergence
SO system initially treats all configurations equally and then it expresses a preference for one possibility. However, there are no objective criteria for preferring one stable configuration over another. The system makes an arbitrary decision to change the range of possibilities. Organizational closure turns a collection of interacting elements into an individual, coherent whole. The whole has properties that arise out of its organization, and that cannot be reduced to the properties of its elements. Such properties are called emergent. Emergence refers to a process by which a system of interacting subunits acquire qualitatively new properties that cannot be fully predicted and cannot be found in the qualities of the components. SO parameter values may change in any generation based on the in-put and significant improvements in results. This unpredictability creates the real novelty.

These system level properties arise unexpectedly from non-linear interactions among a system's components. In the terminology of a dynamic system this emergent pattern or property is called an attractor of the system. Under a particular set of initial conditions and parameter values, an attractor is the state toward which the system converges over time [1, 2, 21, 22].

5. EXISTING MSA METHODS

The rapid accumulation of DNA sequences during last decade made sequence analysis vital in research. MSA, aligning three or more nucleotide or amino acid sequences simultaneously is one of the important tasks in bioinformatics. Important application of MSA is their incorporation in many structure and function prediction methods.

The computation of an optimal alignment mathematically is too complex. Various algorithms available for MSA are classified into three main categories: Exact, Progressive and Iterative based on their properties.

Exact algorithms are of 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 uses dynamic programming and 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 significant 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 [23].

The widely used MSA tools implementing different algorithms are ClustalW [24], MultAlin [25], DIALIGN [26], MUSCLE [27], T-Coffee [28], DCA [29]. In addition GA based MSA softwares SAGA [30], MSA-GA [31] and literatures [6, 32] are also available.

Advantages of GA

- Its flexibility in assigning fitness function, a mathematical function is used to evaluate the fitness of chromosomes.
- GA is efficient in solving NP-hard (non-deterministic polynomial) problems [27]. The complexity of MSA increases exponentially, hence NP-hard.
- It is not restricted to depend on a particular algorithm to solve the problems. Needs only fitness function to evaluate the chromosomes [32].

6. SOGA FOR MULTIPLE SEQUENCE ALIGNMENT

The chromosomes for the multiple sequence alignment process consist of gap positions which are altered to produce better alignments [33]. The chromosome length can be self-organized based on the length of input sequences [6]. The crossover and mutation operator can be made to perform a self-organized selection of crossover/ mutation point and the corresponding rate from the initial crossover/ mutation point. The crossover and mutation operation can be made to undergo a conditional increase in the rates [34]. Thus the process continues for a range of rates cyclically until an optimal upper limit is reached. The number of generations and population size can also be self-organized based on the betterment of the fitness value obtained in the previous generation [35].

Table 1. Comparison of parameters of existing self-organizing genetic algorithms

Algorithm	Population Size	Generation	Selection Operator	Crossover Operator	Crossover Rate	Mutation Operator	Mutation Rate
SOFNNGAPSO [36]	50	5000	Roulette with Elitism	Two point	0.5	Random	0.02
SOGA-MSA [34]	100	SO	Elitism	SO-one point	Varies from initial 0.7	Cyclic	1-80
C-SOMGA [37]	20	occurrence of success run or 150000 function evaluations	Tournament	Single point	0.85	Bitwise	0.005
SOGA [8]	50	100	Dominant and Elitism	One-point	0.7	Cyclic	Periodically varies
SORIGA [38]	12	100	New replacement and selection scheme	Two point	0.6	Bitwise	0.01

7. COMPARATIVE STUDY ON EXISTING SOGA VARIANTS

Several variants of SOGA proposed to improve the performance, mostly in the form of designing the new operators or hybrid algorithms combined with conventional GA. Modified algorithms bring about improvement by addressing premature convergence or diversity in the population. Variants of GA aimed at promoting diversity in the process were studied and the observations were discussed. SOFNNGAPSO is a self-organizing fuzzy neural network based on GA and PSO in which the parameters of the

consequent parts were obtained using the error function and the parameters of the premise parts in an iterative process. GA and PSO were applied to conduct fine tuning for the parameter set of the premise parts and consequent parts in fuzzy

model [36]. The self-organizing genetic algorithm applied to solve multiple sequence alignment (SOGA-MSA) 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. Reducing run time and avoiding premature convergence were observed as the main advantage of the algorithm [34]. C-SOMGA, self-organizing migrating genetic algorithm for constrained optimization is based on the features of genetic algorithm (GA) and self-organizing migrating algorithm (SOMA) [37]. This algorithm is inexpensive in terms of functional evaluation as it involves less population size. Self-organizing genetic algorithm (SOGA) implements a new dominant selection operator that enhances the action of the dominant individuals and a cyclical mutation operator which modify the mutation probability periodically in accordance with evolution generation. SOGA possess good global search property with a high convergence speed [8]. SORIGA propose a new selection scheme and replacement of individuals with the lowest fitness of the current population new randomly generated individuals. Various parameter settings such as, population size, generation, selection operator and crossover/ mutation rate are modified depending on the problem. Implementation of self-organizing behavior in GAs was found to be beneficial for their performance under dynamic environments [38]. In self-organizing genetic algorithm (SOGA) as a multimodal optimizer, various GA parameters such as, population size, crossover/ mutation rate were adaptively assigned during execution which in turn reduce the execution time significantly [39].

8. SHORTCOMINGS OF SOGA

Though there are no major shortcomings that are specifically attributed to the SOGA, there are a few minor issues such as requirement of adequate planning and organization for mapping SO properties to GA. Complete knowledge about the problem to be solved and choosing suitable operators are necessary.

9. CONCLUSIONS

The genetic algorithm with self-organizing coding, operators and parameter values is efficient and simple to use. An attempt towards self-organizing genetic algorithm requires a complete understanding about the relationship among various parameters and their impact on the performance. In conventional GAs, optimal parameter value for a particular problem can be found, by executing the algorithm with all possible values and all possible combinations with other parameter values. This process of optimization requires more time. This time requirement is eliminated in SOGA. SOGA can also be made to undergo learning process, so that time required to solve similar type of problems in future can be further reduced.

The default parameter values assumed to be optimal is fixed in the condition, when user fails to select appropriate values. Even this default optimal value may lead to bad results for some problems. In self-organized GA, almost all or most of the parameters are self-organized based on the problem instead of obtaining from the user. For conventional GAs, user with complete knowledge about the problem can decide which coding method and operators to be used and set the most appropriate values for the parameters. This contradicts to Holland's goal that GA is a robust and easy to use method. In SOGA without a complete knowledge about the problem, the user can obtain the optimal solution. As the search progresses the SOGA adapt the suitable/ appropriate parameter values. Self-organization made GA easy to use, which is the ultimate goal of Holland. This survey is carried on, for developing a SOGA for multiple sequence alignment. With the strategies of GA shown in this paper one can develop self-organizing genetic algorithm for any problem.

10. ACKNOWLEDGMENTS

We acknowledge the support of the Department of Information Technology (DIT) [Ref. No: DIT/R&D/BIO/15(8)/2011], Government of India, New Delhi.

11. REFERENCES

[1] Camazine, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraulaz, J. G., Bonabeau, E. 2001. Self-Organization in Biological Systems. Princeton University Press: New Jersey.

[2] Heylighen, F. 2002. The Science of Self-organization and Adaptivity. In: Knowledge Management, Organizational Intelligence and Learning, and Complexity; L. D. Kie, Eds. In: The Encyclopedia of Life Support Systems (EOLSS). Eolss Publishers: Oxford.

[3] Seeley, T. D. 2002. When Is Self-Organization Used in Biological Systems?. Biol. Bull. 202(3), 314–18.

[4] Jain, L. C., Karr, C. L. 2000. Introduction to evolutionary computing technique. In: Proceedings of the Electronic Technology Directions to the Year 2000. 1995 May 23-25. Adelaide. SA, 122-27.

[5] Marczyk, A. 2004. Genetic Algorithms and Evolutionary Computation. The TalkOrigins Archive. 23 Apr. 2004. http://www.talkorigins.org/faqs/genalg/genalg.html

[6] Wu, S., Lee, M., Gatton, T. M. Multiple Sequence Alignment using GA and NN. International Journal of Signal Processing, Image Processing and Pattern Recognition: 21-30.

[7] Hong, T., Wang, H., Lin, W., Lee, W. 2002. Evolution of Appropriate Crossover and Mutation Operators in a Genetic Process. Applied Intelligence. 16, 7-17.

[8] Zhang, J., Zhuang, J., Du, H., Wang S. 2009. Self-organizing genetic algorithm based tuning of PID controllers. Information Sciences. 179 (7), 1007-18.

[9] Breukelaar, R., Bäck, T. 2008. Self-Adaptive Mutation Rates in Genetic Algorithm for Inverse Design of Cellular Automata (July 2008), 12-6.

[10] Thierens, D. 2002. Adaptive mutation rate control schemes in genetic algorithms. Institute of Information and Computing Sciences. Utrecht Univerisity. The Netherlands.

[11] Bao-Juan, H., Jian, Z., De-Hong, Y. 2008. A Novel and Accelerated Genetic algorithm. WSEAS Transactions on Systems and Control. 3(4), 269-78.

[12] Kubota, N., Fukuda, T., Shimojima, K. 1996. Virus-evolutionary genetic algorithm for a self-organizing manufacturing system. Computers and Industrial Engineering. 30(4), 1015-26.

[13] Ray, S. S., Bandyopadhyay, S., Pal, S. K. 2005. New Genetic Operators for Solving TSP: Application to Microarray Gene Ordering. Springer-Verlag Berlin Heidelberg, 605–610.

[14] Soper, R., Taylor, D. J., Green, N. P. O., Stout, G. W. 1997. Biological Science. 3rd ed. Cambridge University Press: United Kingdom.

[15] Introduction to genetic algorithm. www.obitko.com/tutorials/genetic-algorithms

[16] Genetic Server/ Library product: Neuro Dimension inc. www.nd.com/products/genetic

[17] Evolutionary Algorithm. http://www.geatbx.com/docu/algindex-02.html

[18] Rocha, L. M. Modeling evolution: evolutionary computation. Lecture notes, Biologically Inspired Computing. School of Informatics. Indiana University. http://informatics.indiana.edu/rocha/i-bic/

[19] Harik, G. R., Lobo, F. G. 1999. A Parameter-Less Genetic Algorithm. IEEE Transactions on Evolutionary Computation, 523-8.

[20] Sivanandam, S. N., Deepa, S. N. 2008. Introduction to Genetic Algorithms. Springer: New York.

[21] Kelso, J. A. S. 1995. Dynamic patterns: the self-organization of brain and behavior. MIT Press: USA.

[22] Fuchs, C. 2008. Internet and society: social theory in the information age. Routledge. New York.

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

[24] Thompson, J. D., Higgins, D. G., Gibson, T. J. 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting position specific gap penalties and weight matrix choice. Nucleic Acids Res. 22, 4673-80. http://www.ebi.ac.uk/Tools/clustalw/

[25] Corpet, F. 1988. Multiple sequence alignment with hierarchical clustering. Nucleic Acids Res. 16, 10881-90. http://bioinfo.genotoul.fr/multalin/multalin.html

[26] Morgenstern, B., Dress, A., Wener, T. 1996. Multiple DNA and protein sequence based on segment-to-segment comparison. Proc. Natl. Acad. Sci. 93, 12098-103. http://bibiserv.techfak.uni-bielefeld.de/dialign/

[27] Edgar, R. C. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 32, 1792-7. http://www.ebi.ac.uk/Tools/muscle/

[28] Notredame, C., Higgins, D. G., Heringa, J. 2000. T-Coffee: A novel method for fast and accurate multiple sequence alignment. J Mol Biol. 302, 205-17. http://www.ebi.ac.uk/Tools/t-coffee/

[29] Stoye, J., Moulton, V., Dress, A. W. 1997. DCA: an efficient implementation of the divide-and conquer approach to simultaneous multiple sequence alignment. Comput. Appl. Biosci. 13(6), 625-6. http://bibiserv.techfak.uni-bielefeld.de/dca/

[30] Notredame, C., Higgins, D. G. 1996. SAGA: sequence alignment by Genetic algorithm. Nucleic Acids Res. 24(8), 1515-24.

[31] Gondro, C., Kinghorn, B.P. 2007. A simple Genetic Algorithm for multiple sequence alignment. Genet. Mol. Res. 6 (4), 964-82.

[32] Karadimitriou, K., Kraft, D. H. 1996. Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology. In Proc. 2nd Annual Molecular Biology and Biotechnology Conference, Baton Rouge, LA.

[33] Amouda, V., Selvaraj, V., Kuppuswami, S., Buvaneswari, S. 2010. iMAGA: Intron Multiple Alignment Using Genetic Algorithm. International Journal of Engineering Science and Technology. 2(11), 6361-6370.

[34] Amouda, N., Buvaneswari, S., Kuppuswami, S. 2011. Self-Organizing Genetic Algorithm for Multiple Sequence Alignment. Global Journal of Computer Science and Technology. 11(7), 7-14.

[35] Amouda, V., Buvaneswari, S., Kuppuswami, S. 2011. Self organizing algorithm for multiple sequence alignment. Online Journal of Bioinformatics. 12(1), 74-84.

[36] Khayat, O., Ebadzadeh, M. M., Shahdoosti, H. R., Rajaei R., Khajehnasiri, I. 2009. A novel hybrid algorithm for creating self-organizing fuzzy neural networks. Neurocomputing. 73, 517-524.

[37] Deep, K., Dipti. 2008. A self-organizing migrating genetic algorithm for constrained optimization. Applied Mathematics and Computation. 198, 237-250.

[38] Tinos, R., Yang, S. 2007. A self-Organizing Random Immigrants Genetic Algorithm for Dynamic Optimization Problems. Genetic Programming and Evolvable Machines. 8(3), 255-286.

[39] Jeong, I. K., Lee J. J. 1998. A self-organizing genetic algorithm for multimodal function optimization. Artif. Life Robotics. 2, 48-52.