Genetic Algorithm: A Tutorial Review

Article · October 2009

CITATIONS
35

READS
2,818

6 authors, including:

Farkhod Alisherov
Tashkent University of Information Technology
23 PUBLICATIONS **416** CITATIONS

SEE PROFILE

Debnath Bhattacharyya
K L University
298 PUBLICATIONS **1,729** CITATIONS

SEE PROFILE

International Journal of of Grid and Distributed Computing
Vol.2, No.3, September, 2009

Genetic Algorithm: A Tutorial Review

Deep Malya Mukhopadhyay[1], Maricel O. Balitanas[2], Alisherov Farkhod A.[2], Seung-Hwan Jeon[2], and Debnath Bhattacharyya[1]

[1]*Heritage Institute of Technology, Kolkata-700107, India*
{deepmalya.kolkata,debnathb}@gmail.com[1]

[2]*Hannam University, Daejeon, Korea*
{jhe_c1756,sntdvl}@yahoo.com, jeonseung66@hotmail.com

Abstract

Generally speaking, genetic algorithms are simulations of evolution, of what kind ever. In most cases, however, genetic algorithms are nothing else than probabilistic optimization methods which are based on the principles of evolution. This tutorial covers application oriented study of genetic algorithms as in the case of Eye Location Using Genetic Algorithm; Using simulation and Genetic Algorithms to improve cluster tool performance; Mooring Pattern Optimization using Genetic Algorithms. This tutorial is designed to cover a few important applicational aspects of genetic algorithm under a single umbrella.

Keywords: *Eye Location, genetic algorithm, fuzzy, evolutionary..*

1. Introduction

Knowledge-based information systems or Evolutionary computing algorithms are designed to mimic the performance of biological systems. Evolutionary computing algorithms are used for search and optimization applications and also include fuzzy logic, which provides an approximate reasoning basis for representing uncertain and imprecise knowledge. As the name implies, artificial neural networks mimic the brain or biological information processing mechanisms. These they do in a very limited sense. The no free lunch theorem states that no search algorithm is better on all problems. Know from this no free lunch theorem that all search methods show on average the same performance over all possible problem instances. The present trend is to combine these fields into a hybrid in order that the vagaries of one may be offset by the merits of another. Neural networks, fuzzy logic and evolutionary computing have shown capability on many problems, but have not yet been able to solve the really complex problems that their biological counterparts can. Some of these hybrid techniques are,

Evolutionary algorithm parameters (population size, selection, etc.) controlled by fuzzy systems,

a. Neural network parameters (learning rate) controlled by fuzzy systems,
b. Fuzzy logic controllers generated and tuned by evolutionary algorithms,
c. Fuzzy logic controllers tuned by neural networks,
d. Evolutionary computing in automatically training and generating neural network architectures.

2. Basic steps of a Genetic Algorithm

Figure 1 shows where the field of genetic algorithms is placed in the hierarchy of knowledge based information systems or evolutionary computing. This tutorial considers only genetic algorithms, a branch of evolutionary algorithm that is widely used.

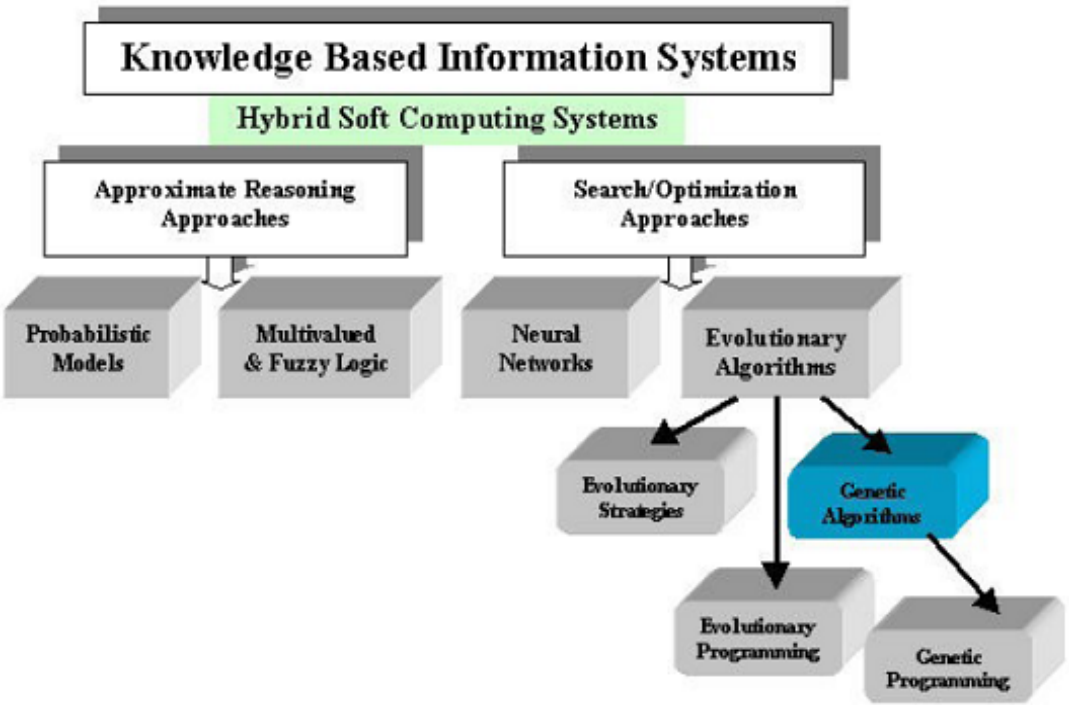

Figure 1. Steps of Genetic Algorithm.

Table 1 gives a list of different expressions, which are common in genetics, along with their equivalent in the framework of GAs:

Table 1. GA Expressions.

Natural Evolution	Genetic Algorithm
genotype	coded string
phenotype	uncoded point
chromosome	string
gene	string position
allele	value at a certain position
fitness	objective function value

Basic structure of a genetic algorithm is given hereunder:
Algorithm
t := 0;
Compute initial population B_0;
WHILE stopping condition not fulfilled DO
BEGIN

Select individuals for reproduction;
Create offspring's by crossing individuals;
Eventually mutate some individuals;
Compute new generation
END

As obvious from the above algorithm, the transition from one generation to the next consists of four basic components:

Selection: Mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).

Crossover: Method of merging the genetic information of two individuals; if the coding is chosen properly, two good parents produce good children.

Mutation: In real evolution, the genetic material can by changed randomly by erroneous reproduction or other deformations of genes, e.g. by gamma radiation. In genetic algorithms, mutation can be realized as a random deformation of the strings with a certain probability. The positive effect is preservation of genetic diversity and, as an effect, that local maxima can be avoided.

Sampling: Procedure which computes a new generation from the previous one and its off springs.

Compared with traditional continuous optimization methods, such as Newton or gradient descent methods, we can state the following significant differences:

a. GAs manipulate coded versions of the problem parameters instead of the parameters themselves, i.e. the search space is S instead of X itself.
b. While almost all conventional methods search from a single point, GAs always operates on a whole population of points (strings). This contributes much to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and, vice versa, reduces the risk of becoming trapped in a local stationary point.
c. Normal genetic algorithms do not use any auxiliary information about the objective function value such as derivatives. Therefore, they can be applied to any kind of continuous or discrete optimization problem. The only thing to be done is to specify a meaningful decoding function.
d. GAs use probabilistic transition operators while conventional methods for continuous optimization apply deterministic transition operators.

More specifically, the way a new generation is computed from the actual one has some random components (we will see later by the help of some examples what these random components are like).

3. Analysis of existing Algorithms

3.1 Eye Location Using Genetic Algorithm

This algorithm [3] is designed with the view of being a sub branch of automatic face recognition. The adaptive eye location approach seeks first where salient things are and then what their identity is. Specifically, eye location involves (i) the derivation of the saliency attention map, and (ii) the possible classification of salient locations as eye regions. The saliency ('where') map is derived using consensus between navigation routines encoded as finite state automata (FSA) exploring the facial landscape and evolved using genetic algorithms (GAs). The classification ('what') stage is concerned with the optimal selection of

features and the derivation of DT (decision trees) for confirmation of eye classification using genetic algorithms (GA). Figures 2 and 3, assist in defining the approach.

The resolution of input face image is 192x192 using 8 bit gray levels. To account for illumination changes, the original images are processed using 5x5 Laplacian masks. The Laplacian filters out small changes due to illumination, and detects those image transitions usually associated with changes in image contents or contrast. Three feature maps corresponding to the mean, standard deviation (s.d.) and entropy are then computed over 6x6 non-overlapping windows and then compressed using quantization to yield three 32x32 feature maps, encoded using four gray levels (2 bits) only. The original facial images are also resized to the resolution of 32x32 for display purposes. Examples of such feature maps are shown in figure 4.

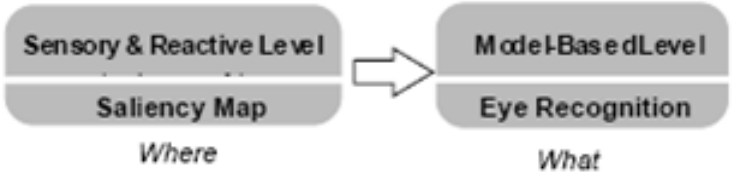

Figure 2. Architecture for eye location.

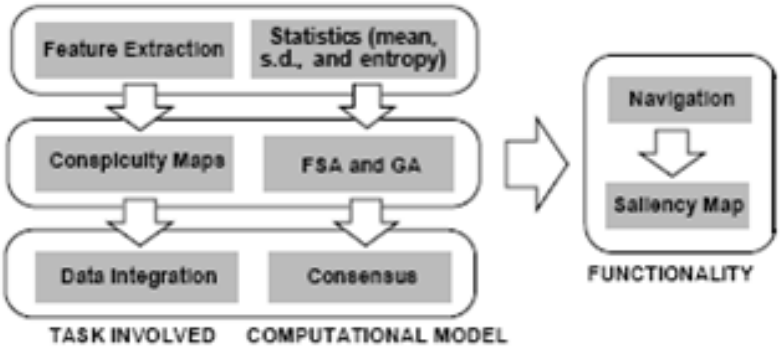

Figure 3. Derivation of the saliency map.

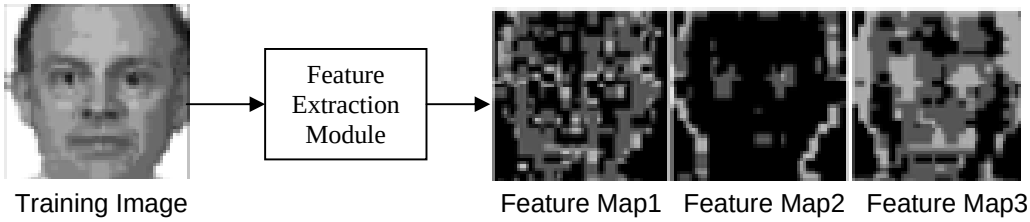

Figure 4. Feature maps.

The algorithm uses eye location as a test bed for developing navigation routines implemented as visual routines (VR) driven by an adaptive behavior-based AI. While there are many eye location methods our technique is the first to approach such a location task using navigational routines and to automate the derivation of such routines using evolution

and learning rather than manually handcrafting them. The adaptive eye location approach seeks first where salient things are and then what their identity is.

3.2 Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance

The proposed algorithm combines simulation and a genetic algorithm to generate lot processing sequences. The Simulation Engine and Cluster Tool model forms the main backbone of the proposed algorithm [1].

Simulation Engine: In order to perform the simulation studies, a simulation engine for cluster tools was developed in C++. The simulation model can consist of an arbitrary number of cluster tools, each of which can have an arbitrary number of process chambers, load locks and handlers. The most important parameters that can be specified in the simulation model are listed in Table 2.

Table 2. Model Parameters.

Cluster tools:	- Number of process chambers - Number of load locks - Number of handlers
Handlers:	- Move time (with or without wafer) for every origin/ destination pair
Load locks:	- Pump/vent time
Sequences:	- Number of process steps
Process steps:	- Process chambers qualified for a step - Process time in these chambers
Lots:	- Number of wafers per lot
Wafers:	- Process sequence

For each of the components of a cluster tool, down times can be specified. However, for this study, down times were not incorporated in the model. The simulation engine computes a number of output statistics after a simulation run, such as cycle times of wafers and utilization of the components of a cluster tool. For the studies described in this paper, the total completion time for a given sequence of lots is of special interest. The modular concept of the simulation tool allows exchanging different parts of the model and testing their impact on performance. For example, the module responsible for the control of the handler can currently be chosen from a set of four modules: a FIFO based control, a Least Slack based control, and a Critical Ratio based control and a module that uses backtracking to find optimal wafer moves.

Cluster Tool Model: The cluster tool model under investigation in this paper is depicted in Figure 5. It consists of two main modules to which the individual processing chambers are attached. Transportation of wafers in the upper module is done by the transfer robot, in the

lower module the wafers are transported by the buffer robot. There are two load locks that allow load lots into the cluster tool independently and process them in parallel

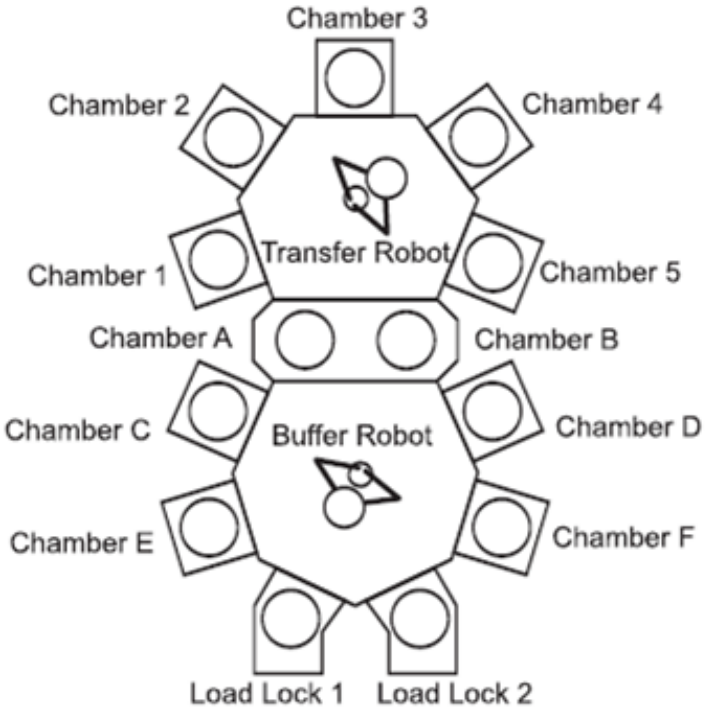

Figure 5. Cluster Tool Model.

The model parameters are as follows.

For both robots, it is assumed that it takes 20 seconds to move a wafer from any position (chamber or load lock) to another. Without transporting a wafer, it takes the robots one second to move from one position to another. Pump and vent times for the load locks are zero, since they were not regarded in the study. A simulation study for this model was conducted for as many as 30 different process sequences. In this paper an approach was presented that uses a simulation model of a set of cluster tools and a genetic algorithm (GA) to find optimal processing sequences of lots at these cluster tools. Several sample applications showed that the proposed method can be used to produce optimal or close–to–optimal sequences in short time. When applied on the production floor, this algorithm can lead to a significant reduction in cycle times.

3.3 Mooring Pattern Optimization using Genetic Algorithms

Authors presented the development of a Genetic Algorithm (GA) to solve the problem of the mooring pattern of floating units used in oil exploitation operations. The distribution of mooring lines is one of the factors that directly influence the displacements (offsets) suffered by floating units when subjected to environmental conditions such as winds, waves and currents. In this paper, GA seeks an optimum distribution of the mooring lines whose final goal is to minimize the units displacements. The computation of the floating unit's static equilibrium position is accomplished by applying the catenary equilibrium equation to each

mooring line in order to obtain the out-of-balance forces on the unit, and by using an iterative process to compute the final unit equilibrium position [2].

Mooring Systems Design: There are several factors to be considered when designing a mooring system, such as the type of anchorage, the strength of the mooring lines, the position of the anchors, the different sets of environmental conditions the unit will be subjected to, the seabed's shape and the time the floating unit will remain anchored. Mooring lines are usually composed of different types of materials, each with different physical and mechanical properties. Buoys and/or clump weights may be attached to them in order to deal with the restriction imposed either by other mooring systems or by the seabed's layout.

As in most engineering problems, cost vs. effectiveness is certainly an issue in the mooring system design process. Selecting materials, defining the environmental conditions to be imposed on the floating unit and any other choices will influence directly both cost and effectiveness. A typical example of a catenary mooring system is illustrated in Figure 6.

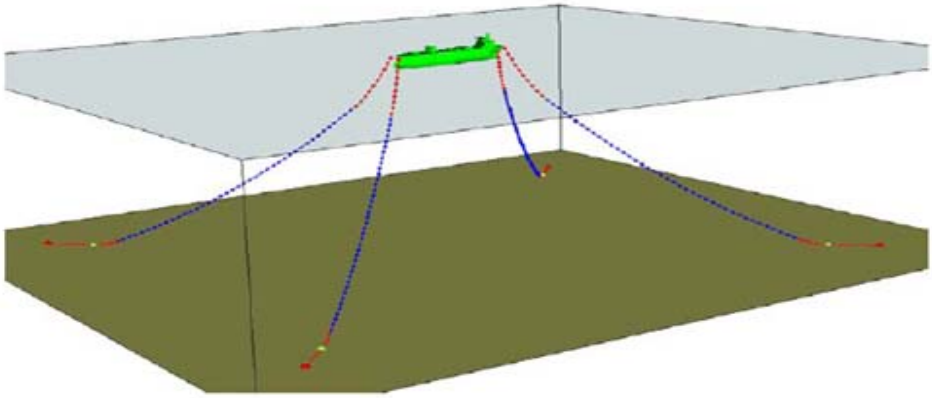

Figure 6. Catenary mooring system.

The main computational steps of the proposed algorithm are enlisted in the table 3.

A proposed genetic algorithm to solve mooring pattern optimization problems was presented. The main feature of this algorithm is the substitution of only one or two individuals per generation. The ranking selection technique was adopted.

4. Discussion

4.1 Eye Location Using Genetic Algorithm

This algorithm was the pioneer to approach eye location task in a facial image using navigational routines and to automate the derivation of such routines using learning and evolution rather than manually handcrafting them.

After going through the algorithm, it was my observation that FSA (finite state automata) used for exploring the facial landscape and evolved using genetic algorithm has its starting point fixed at the center of the chin. But as the algorithm is designed to track the location of the eye, The algorithm will process much faster ensuring higher reliability and efficiency if the starting point of FSA exploration is fixed at the center of forehead instead of center of chin. When the navigational pointer starts from center of forehead it will explore the location of eye in the facial landscape much faster than if it had started from the center of chin.

4.2 Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance

An approach was attempted to generate optimal processing sequences of lots at cluster tools. It considers the problem of sequencing n lots, where each lot can be processed by any of m available cluster tools. The proposed method combines simulation and a genetic algorithm to generate lot processing sequences. The algorithm claims that it leads to a significant reduction of cycle times at cluster tools.

According to observation, two possible improvements can be suggested to the algorithm. For instance, if a GA-within-GA is used to generate lot processing sequence instead of the combination of simulation engine and genetic algorithm, it might lead to a better performance.

Secondly we can stick to the proposed algorithm provided that the number of cluster tools, number of process chambers in each cluster tools, number of load locks and number of handlers are all taken to be fixed and made application specific instead of arbitrary numbers that they are in the proposed algorithm. This might lead to significant reduction of cycle times at cluster tools.

4.3 Mooring Pattern Optimization using Genetic Algorithms

This paper makes use of Genetic Algorithm (GA) to solve the problem of the mooring pattern of floating units used in oil exploitation operations. Mooring lines are used for positioning of floating units during oil extraction with a goal of minimizing the displacement suffered by floating units.

According to my understanding the proposed algorithm can be more effective if the following things can be taken care of: a) the algorithm is very much dependent on natural parameters b) Application of Genetic algorithm and its utilization can be increased.

5. Conclusion

The main objective of the tutorial paper was to throw some light on the applicational aspect of the Genetic Algorithm in different domains, which may be real-life, or hypothetical in nature. Research in The Genetic Algorithm is progressing very fast and numerous researchers from various fields are focusing to develop some workable scheme. Different companies are also working to get commercial products. Hoping for some commercial and effective schemes that will be available in future.

Acknowledgement

This work was supported by the Security Engineering Research Center, granted by the Korea Ministry of Knowledge Economy..

References

[1] Mathias A. Dummler, “Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance”, The 1999 Winter Simulation Conference, December 5 – 8, 1999, Squaw Peak, Phoenix, AZ, pp. 875-879.
[2] Alonso J. Juvinao Carbono, Ivan F. M. Menezes, Luiz Fern, O Martha, “Mooring Pattern Optimization using Genetic Algorithms”, 6th World Congresses of Structural and Multidisciplinary Optimization Rio de Janeiro, 30 May - 03 June 2005, Brazil. http:// www.wcsmo6.org/papers/882.pdf
[3] Jeffrey Huang and Harry Wechsler, “Eye Location Using Genetic Algorithm”, 2nd International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA), March 22-23, 1999, Washington, DC, USA.