Adaptive Genetic Algorithms Based on Learning Classifier Systems

R. Shamsaei, A. Hamzeh, A. Rahmani
{shamsaei, hamzeh, rahmani}@iust.ac.ir
Department of Computer Engineering
Iran University of Science and Technology
Tehran, Iran

Abstract

Genetic Algorithms (GA) emulate the natural evolution process and maintain population of potential solutions to a given problem. But GA uses static configuration parameters such as crossover type, crossover probability and selection operator, among those, to emulate this inherently dynamic process. Because of dynamic behavior of GA and changes in population parameters in each generation, using adaptive configuration parameters sounds a good idea. This idea is considered in some researches about GA [1, 2, 3, and 4] by various authors. In this research a new modified structure for GA is introduced which called Adaptive GA based on Learning classifier systems (AGAL). AGAL uses a learning component to adapt its structure as population changes. This learning component uses domain knowledge which is extracted from the environment to adapt GA parameter settings.

Keywords: Genetic Algorithms, Learning Classifier Systems, Crossover Operators, Adaptive Genetic Algorithms

1. Introduction

GA Behavior is strongly determined by the balance between exploiting what already works best and exploring possibilities that might eventually evolve into something better. The loss of critical alleles due to selection pressure, the selection noise, the schemata disruption due to crossover operator, and poor parameter setting may make this exploitation/exploration relationship disproportionate and guide process to the trap of the premature convergence problem. However, finding robust genetic operators or parameter settings that allow the premature convergence problem to be avoided in any problem is not a trivial task, since their interaction with GA performance is complex and the optimal ones are problem dependent. Furthermore, different operators or parameter configurations may be necessary during the course of a run for inducing an optimal exploitation/exploration relationship. For these reasons, many adaptive techniques have been suggested for changing the GA configuration depending on parameters associated with the GA performance, in order to offer the most appropriate exploitation/exploration behavior [5, 6,

7, 8, and 9]. Adaptive methods may be categorized based on the GA components that are adapted throughout the run: 1) Adaptive parameter settings, 2) Adaptive genetic operator selection, 3) Adaptive genetic operators 4) Adaptive representation and 5) Adaptive fitness function [4]. The goal of this paper is to introduce an adaptive genetic algorithm based on adaptive operator selection and adaptive parameter settings, this adaptation is achieved by using a Learning Classifier System [10] as an expert to gather domain knowledge about GA behavior and exploit them in parameter setting and operator selection.

2. Adaptive Genetic Algorithm

Now, we report on some adaptive approaches that are described in the literature for adjusting the GA control parameters such as mutation probability, crossover probability and population size. In [5], an adaptive method for mutation probability is used. Its principle features are: 1) each position of each chromosome has an associated mutation probability; 2) these probabilities are incorporated into genetic representation of the chromosomes encoded as bit strings and 3) they undergo evolution as well as the chromosomes. In [6], a GA with varying population size was discussed. When a chromosome is born a lifetime is assigned to it, the death of this chromosome occurs when its age exceeds its lifetime value. In the calculation of the lifetime value the current state of the GA is taken into consideration. In [7], an adaptive GA was proposed, where crossover probability and mutation probability are varied depending on the fitness values of the solution. Next we review different proposals of adaptive operator selection. In [8], a crossover mechanism was proposed wherein the distribution of crossover points, at which crossover is allowed to cut and splice material, is adapted. This mechanism attaches to each individual a binary encoded description of how to perform crossover on this individual. In [9] an attempt very similar to pervious one is represented which only allows tow type of crossover and represent them with one attached bit to the end of each chromosome.

3. An Adaptive GA Based on LCS (AGAL)

In this section, we explain our proposed method. In this method, we use a Learning Classifier System (LCS) to adapt GA parameters. Hence, we call our GA, Adaptive GA based on LCS, or AGAL. AGAL structure is shown in Fig. 1. To run this process we have used tournament selection [11] with dynamic tour size. Crossover type is initially set to uniform crossover [12] with probability of 0.5 (P0=.5). These parts are basic parts of traditional GA. For more information about their task and structure refer to [13]. The main part of this algorithm is VSCS, which is a learning Very Simple Classifier System [14]. This part is used to coordinate and guide the entire evolution process. The VSCS learns some rules about population and sets some parameters to run AGAL. These parameters include: selection tour size (which fed to selection part), Crossover type (which toggled between one-point and uniform crossover) and probability of uniform crossover (P0), which is required for uniform crossover. These parameters are fed to crossover part. The last parameter is mutation rate which is fed to mutation part. To do this task, coordinator receives some population parameters (which are described later) and produce above parameters to adapt AGAL process with population evolution through entire process. These parts are explained in more detail in the following sections. (Initial values to start process, are Tour size=2, Crossover rate=0.8 and Mutation rate=0.01)

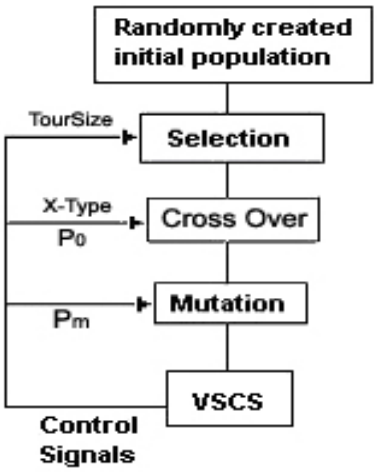

Fig. 1: AGAL structure.

4. Test Functions

To test AGAL performance, two functions are selected from previous studies [15]. One of them is a unimodal function with only one minima and convex shape. Unimodal function is $f_1(x_1,x_2) = (x^2_1+x_2-11)^2+(x_1+x^2_2-7)^2$ in the feasible region of $0\le x_1,x_2\le 6$. It has a minimum solution at (x_1=3, x_2=2) and is shown in Fig. 2.

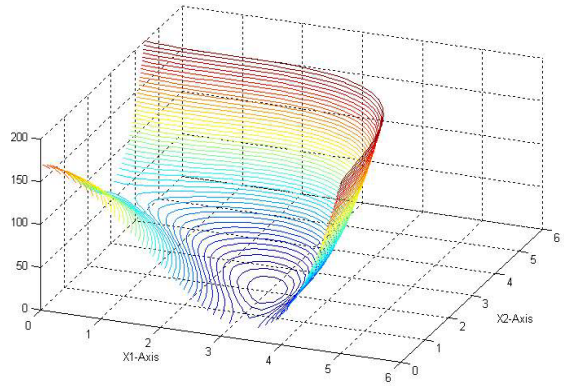

Fig2.

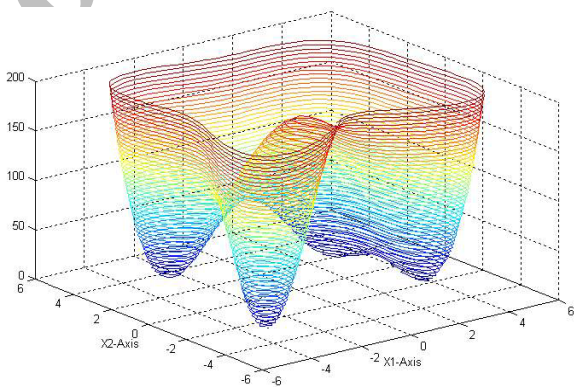

Fig. 3

Another one is a multimodal function which has four minima and one global minimum. Multimodal function is: $f_2(x_1,x_2) = (x^2_1+x_2-11)^2+(x_1+x^2_2-7)^2+0.1[(x_1-3)^2+(x_2-2)^2]$ in the feasible region of $-6\le x_1,x_2\le 6$. Fig. 3 shows the contour plot of this function. The representation used for both functions is binary representation with a string length of 128 bits, 64 bits for each variable. Hence, the size of search space is over 4 billions points.

5. Termination conditions and performance criterion

Termination conditions for both of these functions are as follows:

1- Best individual fitness will be in neighborhood (± 0.002) of optimum solution.
2- To reach a maximum generation of 100 for the first function and 200 for the second one.

Performance criterion is average fitness of population per each generation.

6. Very Simple Classifier System (VSCS)

VSCS is a simple LCS which introduced in [14], VSCS is a learner which based on condition-action rules and is used in simple environment with one action per condition where some possible condition-action rules are determined externally and stored in VSCS database. These assumptions are similar to those made by Dorigo and Bersini [14]. Following their notation, a classifier i (rule i) is described by (c_i, a_i), where c_i and a_i are respectively the condition and action parts of the classifier. S_t (c_i, a_i) gives the strength of classifier i at time step t. This time step t is a GA generation. The algorithm is presented below:

```
Initialization
 Create a classifier for each state-action pair;
 t: = 0 ;
 Set St(cc, ac), the strength at time t of classifier c, to an initial value;
          {cc is the condition part of classifier c, while ac is its action part}.

Repeat forever
 Read(m) {m is the sensor message};
 Let M be the matching set;
 Choose the firing classifier c ϵM, with a
 probability given by St(cc,ac)/∑ d ϵ M St(cd,ad);
 Change classifiers strength according to the implicit bucket brigade;
 t:=t+1;
 Execute(ac);
```

Fig. 4: the very simple classifier system (VSCS) [5]

All classifiers are initiated to strength 0.1 in this case and *implicit bucket brigade* algorithm [14] is used to update strength of classifiers according to Equation 1:

$$S_{t+1}(c_c, a_c) = (1- \alpha)\, S_t(c_c, a_c) + R + \alpha\, S_{t+1}(c_d, a_d) \quad (1)$$

S_{t+1} (c_c, a_c) is the strength of classifier c at time t+1, subscripts c and d identify the classifiers to which conditions and actions belong (e.g., c_d is the condition part of classifier d), α is learning rate and R is reward from environment. Equation 1 essentially says that, each time a classifier is activated its strength changes, and that this change amounts to the algebraic sum of outgoing payments (the - α times strength component), environmental rewards, and ingoing payments. The learning rate is chosen 0.9 and R is chosen based on improvement of mode of population fitness.

6.1. Condition-Action parts

Based on VSCS description in [14], each rule contains a simple action part and a simple condition. So we can consider a rule as, ***If Condition then Action.*** Conditions and actions are defined in the following sections:

- Condition parts of classifiers are designed based on four population features:

1-Average fitness of population
2-Standard deviation of population fitness
3-Best fitness of population
4-Mode fitness of population

For each of the above categories two values are defined: *near by* and *far away* (*low* and *high* for standard deviation) they are defined as follows:

1-Near by is true about variable A when A is in a predefined interval:

-3*Standard Deviation of A ≤ A ≤3*Standard Deviation of A

This definition is used for Average, Best and Mode fitness. For example 'Average fitness of population is nearby' is a condition.

2-Far away is true when A does not satisfy above condition.
(Low standard deviation=SD is defined crisply $-0.1 \leq SD \leq 0.1$ and the values that are not in this interval are considered as high)

- Action Parts are designed based on the following five control parameters:
  1-Tour Size
  2- P_0 (used as probability of uniform crossover)
  3-Crossover Rate
  4-Crossover Type
  5-Mutation Rate

  For each of the above parameters, the action part contains one of three commands which are named 'increase', 'decrease' and 'change type'. These commands guide AGAL as follows:

  Increase: This command causes AGAL to increase desired variable at constant rate, for example, if action part was 'increase Tour size' and Tour size was 2, then Tour size will be 3. It is notable that these values must stay on their intervals after changing. These intervals are defined later.
  Decrease: This command causes AGAL to decrease desired variable in the same manner as *Increase* command.
  Change Type: This command toggles between 'Uniform' and 'One point' crossover operators.

  Intervals of the above parameters are defined as follows.

Tour Size	**[2, Maximum population/2]**
P_0	**[0.5, 1]**
Crossover Rate	**[0.5, 1]**
Mutation Rate	**[0.00001, 0.1]**

In consequence of above definitions we have 80* rules by now that all of them are initialized to strength 0.1 equally. Below, there are two sample rules:

IF 'Average fitness of population' is near by THEN decrease Tour Size.
IF Best fitness of population Individual is near by THEN decrease CrossOverRate

7. Results

Results of this experiment are calculated using average fitness of 100 independent runs of AGAL and traditional GA using defined function in section 2.1. In Fig. 5, performance of traditional GA is shown against performance of AGAL. As we can see, AGAL outperforms traditional GA.

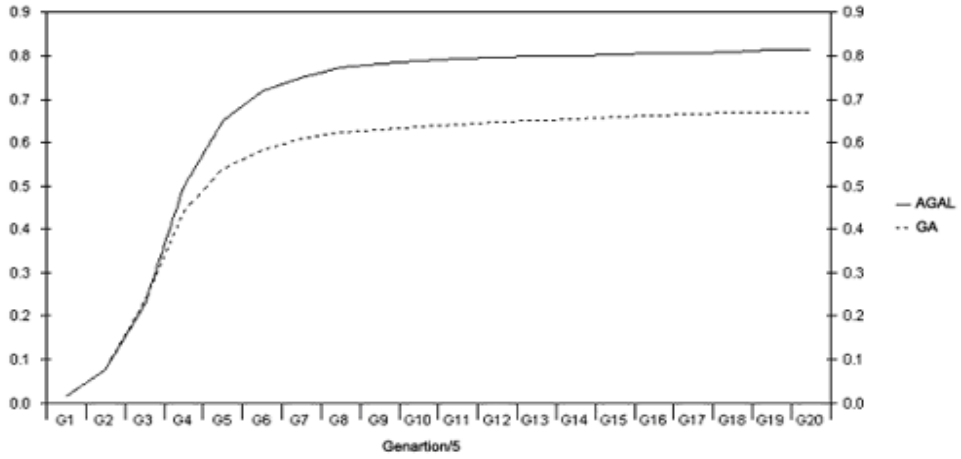

Fig. 5: AGAL vs. GA in Unimodal Test

But in multimodal function, as shown in Fig. 6, AGAL is more successful and shows great performance against traditional GA.

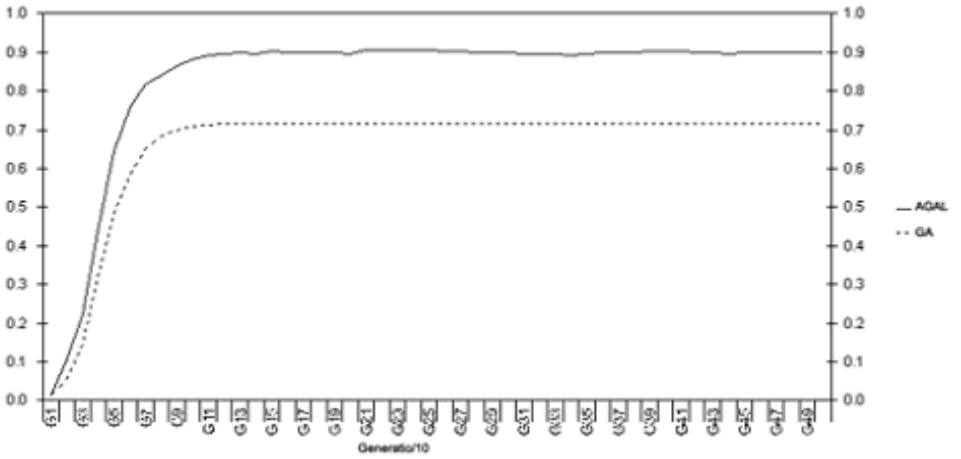

Fig. 6: AGAL vs. GA in Multimodal Test

This figure shows that AGAL adaptive parameter setting, operator selection and adaptive operators in AGAL structure can improve traditional GA performance and convergence speed.

8. Domain knowledge extraction

Using VSCS in AGAL has some other advantages than improving GA performance. As a side effect of VSCS process we have 80 classifiers (rules) that their strength have been tuned throughout of 100 independent runs for each generation. These rules with respect to

their strengths can help other GA users to select best parameter settings and suitable genetic operators (selection and crossover operators) based on AGAL tuned rules strengths, high strength means more suitable rules and vice versa. For example, Fig. 7 shows strength of the classifier *'if Standard Deviation is low then crossover type should change to one point'*. It should be considered that all rules strength are initialized to 0.1 at first, this figure shows that the strength at generation G20 is more than 0.18, this indicated the *importance* of this rule has increased about %15 from first generation and importance of this rule is more than %80 from what we have guessed initially. There are 79 other results that show the rules importance.

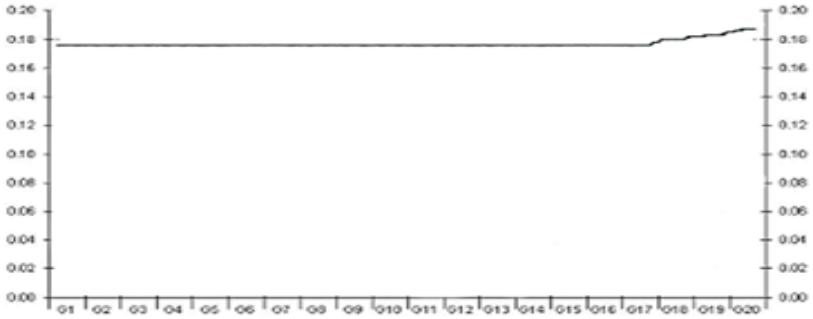

Fig. 7: Strength of classifier: *'if Standard Deviation is low then crossover type should change to one point'*

9. Further Researches

This research has been conducted without considering Fuzzy concepts. It seems that using fuzzy variables which used in classifiers can improve AGAL performance. Herrera and Lozano have done an extensive work around fuzzy [4] and defined many criteria that can be adapted for this research. Changing the VSCS in a form that can control the population size seems to be very helpful. Modifying Selection types by VSCS based on [15] will be helpful too. Defining new ways for extracting knowledge from captured rule strength are useful ways for understanding unknown domains. Another point of view is using XCS [16] instead of VSCS which can extract knowledge itself from the environment and has no need to any initial knowledge.

10. Conclusion

In this paper, we propose a new adaptive GA method, the AGAL that can be used as instead of traditional GAs. This model enables GA to adapt its structure as the population evolves, by extracting domain knowledge from the environment, and hence improving the GA performance. The presented experimental results show that AGAL performs better than traditional GA on the test problems.

11. References:

[1] K. A. De Jong and W. M. Spears 1992, "A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms"

[2] F. Herrera, L. Lozano, J. L. Verdegay, Dynamic and heuristic fuzzy connectives based crossover operators for controlling the diversity and convergence of real coded genetic algorithms. *International journal of intelligent system.* 1996

[3] F. Herrera, L. Lozano, J. L. Verdegay, The use of fuzzy connectives to design real-coded genetic algorithms, 1995,*Mathware & Soft computing 1*(3) 239-251

[4] F. Herrera and L. Lozano, Adaptation of Genetic Algorithm Parameters based on Fuzzy logic controller, 1992, Supported by DGICYT PB92-0933

[5] T. Back. Self-adaptation in genetic algorithms. *Proc. Of the first Int. Conf. on Genetic Algorithms*, 1992, P101-111.

[6] J. Arabas, Z. Michelewicz, J. Mulawka. GAVaPS- a Genetic Algorithms with varying population size. *Proc of The First IEEE Conference on Evolutionary Computation, 1994,* p: 73-78

[7] M. Srinivas, L.M. Patniak. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. On Systems, Man, and Cybernetics*, 1994.

[8] J.D. Schaffer, A. Morshima. An adaptive crossover distribution mechanism for genetic algorithms. *Proc. Second Int. Conf. on Genetic Algorithms*, 1987, p:34-60

[9] W.M. Spears. Adapting crossover in a genetic algorithms. Laboratory report#AIC-92-025, Navy Center for Applied Research in Artificial Intelligence, USA, 1992

[10] P. L. Lanzi, W. Stolzman, and S. W. Wilson, editors, *Advances in Learning Classifiers System*, Springer-verlag, Berlin, 2001

[11] Blickle, T., Thiele, L. A Mathematical Analysis of Tournament Selection, in L.J. Eshelman (ed.) *Proc. 6'Th International Conference on Genetic Algorithms*, Morgan Kaufmann, 1995, pp. 9-16.

[12] Syswerda, G. Uniform crossover in genetic algorithms. In Schaer, J. D., *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, 1989,

[13] Deb. K.,. Genetic Algorithms in search and optimization: The techniques and application. *Proceeding of International Workshop on Soft Computing and Intelligent Systems*, 1998

[14] M. Dorigo and H. Bersini, A Comparison of Q-Learning and Classifier systems *Third International Conference on Simulation of adaptive behavior (SAB94)*, 1994

[15] T. Blickle and L. Thiele, A Comparison of Selection Schemes used in Genetic Algorithms, TIK-Report, Nr.11 Version 2, (2 Edition), 1995

[16] Butz, M. and Wilson, S. An algorithmic description of XCS. In Lanzi, P. L., Stolzmann, W., and S. W. Wilson (Eds.), Advances in Learning Classifier Systems. *Third International Workshop*-IWLCS, 2000.