International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:2, 2008

Statistical Genetic Algorithm

Mohammad Ali Tabarzad, Caro Lucas, and Ali Hamzeh

***Abstract*—**Adaptive Genetic Algorithms extend the Standard Gas to use dynamic procedures to apply evolutionary operators such as crossover, mutation and selection. In this paper, we try to propose a new adaptive genetic algorithm, which is based on the statistical information of the population as a guideline to tune its crossover, selection and mutation operators. This algorithms is called Statistical Genetic Algorithm and is compared with traditional GA in some benchmark problems.

***Keywords*—**Genetic Algorithms, Statistical Information of the Population, PAUX, SSO.

I. Introduction

ADAPTIVE Genetic Algorithms [1] extends Standard Gas to use dynamic procedures to apply evolutionary operators such as crossover, mutation and selection. These methods cause these operators to behave differently with respect to population structure and genetic algorithm's age.

One of the main methods that are used in Adaptive Genetic Algorithms is based on statistical information of the population. It means that we can use this statistical information to alter behavior of crossover, mutation and selection operators to improve their performance. This information can be used in some different manners: for example we can use them to tune involved probabilities such as crossover probability or can be used as a guide line for new operators to act optimally. In our previous researches, we used the later method to produce two new operators: Pattern-based Adaptive Uniform Crossover (*PAUX*) [1] and Stochastic Selection Operator (*SSO*) [2].

In this paper, we are going to propose a new genetic algorithm which is completely based on population's patterns which uses pattern based operators as main evolutionary operators and uses stochastic information of population to drive necessary probabilities such as crossover probability. This algorithm is introduced in this paper and is called "Statistical Genetic Algorithm" or *SGA*. In this paper *SGA* is completely described and compared with traditional *GA* and simple GA which uses *PAUX* or SSO separately in some benchmark problems.

Rest of this paper is organized as follows: in the next sections, we briefly describe some other relevant works on Adaptive *GA*, then we describe *SSO* and *PAUX* in brief and then *SGA* is described in detail. After that, we describe details of benchmark problem selection procedure and benchmark problems, and at last, *SGA* is compared with simple *GA* and *GA* with *PAUX* and *SSO*, the results are then discussed and summarized.

M. A. Tabarzad is with the Center of Excellence: Control and Intelligent Processing, University of Tehran, Tehran, Iran (phone: +98-021-88027757; fax: +98-021-88633029; e-mail: m.tabarzad@ ece.ut.ac.ir).

C. Lucas is with the Center of Excellence: Control and Intelligent Processing, University of Tehran, Tehran, Iran and School of Cognitive Sciences, IPM, Iran (e-mail: lucas@ipm.ir).

A. Hamzeh is with the Computer Engineering Department, Iran University of Science and Technology, Tehran, Iran (e-mail: hamzeh@iust.ac.ir).

II. Relevant Works on Adaptive GA

In this section, some of the most important researches that are recently dome about adaptive genetic algorithms, using adaptive operators, adaptive rates or fuzzy controllers are presented.

In [3], author mentions that through the population, GAs implicitly maintain the statistics about the search space. This implicit statistics can be used explicitly to enhance GA's performance. Inspired by this idea, a statistics-based adaptive non-uniform crossover (SANUX) has been proposed. SANUX uses the statistics information of the alleles in each locus to adaptively calculate the swapping probability of that locus for crossover operation. A simple triangular function has been used to calculate the swapping probability. In this paper two new functions, the trapezoid and exponential functions, are proposed for SANUX instead of the triangular function. Experiment results show that both functions further improve the performance of SANUX

In [4], authors propose a technique for adapting the operator rates, as well as a technique for adapting the parameter values of the operators in a genetic algorithm. They show how these two techniques can be integrated into a single evolutionary system, which is called Integrated-Adaptive Genetic Algorithm (IAGA). The IAGA exibit fewer input parameters to adjust than the original GA, while being able to automatically adapt itself to the particularities of the optimization problem it tackles. They present a proof-of-concept implementation of this technique for royal-road functions and the experimental results.

In [5], author provided an extension of their previous work on adaptive genetic algorithm [6]. Each individual encodes the probability (rate) of its genetic operators. In every generation, only one operator modifies each individual. This operator is selected according to its encoded rates. The rates are updated according to the performance achieved by the offspring (compared to its parents) and a random learning rate. The proposed approach is augmented with a simple transposition operator and tested on a number of benchmark functions.

In [7], authors analyze the Fuzzy Adaptive GAs in depth.

First, they describe the steps for their design and present an instance, which is studied from an empirical point of view. Then, they propose taxonomy for FAGAs, attending on the combination of two aspects: the level where the adaptation takes place and the way the Rule-Bases are obtained. Furthermore, FAGAs belonging to different groups of the taxonomy are reviewed. Finally, they identify some open issues, and summarize a few new promising research directions on the topic. From the results provided by the approaches presented in the literature and the experimental results achieved in this paper, an important conclusion is obtained: the use of fuzzy logic controllers to adapt genetic algorithm parameters may really improve the genetic algorithm performance.

At last, in [8], a fuzzy adaptive search method for genetic algorithms has been proposed, which is able to tune the genetic parameters according to the search stage by the fuzzy rule. In this research, a fuzzy adaptive search method for parallel genetic algorithms is developed, in which the high-speed search ability of fuzzy adaptive tuning by FASGA is combined with the high-quality solution capacity of parallel genetic algorithms. The proposed method offers improved search performance, and produces high-quality solutions. Simulations are performed to confirm the efficiency of the theoretic results, which is shown to be superior to both ordinary and parallel genetic algorithms.

III. Stochastic Selection Operator

In this section, we are going to introduce a pattern based selection operator that was introduced for the first time in [2]. This operator's design is based on two main ideas: it tries to select the best chromosomes in each iteration to carry out them to the next generation, and to preserve genotypic contents and diversity in the population. This operator is called Statistical Selection Operator, SSO.

The main idea behind this operator is to extract hidden genotypic information from the population and to let the entire genetic material of population to survive as long as possible. As mentioned in some basic researches [9], premature convergence is one of the major problems in GA area. This problem relates to the uniformity of the population's genotypic content and inability of evolutionary process to produce new genetic material when it reaches local optima. This problem occurs because after producing some good chromosomes (with respect to their phenotypic material or fitness) they try to distribute their genotypic content using selection and recombination operators. So, other chromosomes have been deleted from the population after some iterations (it depends on their fitness or phenotypic material). In the traditionally GAs, there exist only two solutions to produce new genetic content: using mutation or recombination to produce new chromosome with better fitness than the current best one to carry out the population from current local optima.

Selection operators use fitness as the only criterion to select chromosomes for the next generation. However, in SSO, each chromosome is evaluated using two parameters: fitness and a new measure based on its distance form the best chromosome in the population. We call this measure Genetic Advantage Criterion (GAC).

The main motivation to introduce GAC is to preserve genotypic content of the population. Therefore, it was assumed that if one chromosome has different genotypic content than the other ones in the population, then it is one of the valuable individuals. In the SSO, we compare genotypic structure of that chromosome with genotypic structure of the best individual in the population. To measure their differences, we use Hamming Distance as the measuring tool to compare those chromosomes but it is notable that any other distance function can be used to produce different versions of SSO. However, GAC is calculated using HD(C, Best), where C is current chromosome and Best is the best chromosome of the population.

After calculating GAC, Survival Probability (SP) of each chromosome is calculated using (1):

$$SP = w_1 GAC + w_2 FIT \quad (1)$$

It is clear that SP is a balanced equilibrium between genotypic (GAC) and phenotypic (FITness) evaluation of the current chromosome. It is notable that w1 and w2 are chosen experimentally for each problem and are used to control the importance of genotypic and phenotypic content of the current chromosome. After calculating SP, SSO uses roulette wheel method [9] to produce new generation of chromosomes. It is interesting point that we can use any other selection method for SSO such as tournament or proportionate selection, the only needed modification is to use SP instead of fitness to select winner chromosomes.

IV. Pattern-Based Adaptive Uniform Crossover

In this section, we are going to describe our pattern based operator, that was called PAUX, which is introduced in [1] for the first time. PAUX is designed as an adaptive crossover operator that is based on statistical pattern of the entire population. PAUX is designed using both selection mechanism and crossover methods. It means that in the case of crossover action, both fitness and crossover criteria are considered to produce offspring. PAUX's architecture is based on uniform crossover operator. The Uniform Crossover operator uses a constant called Pu as the probability of bit swapping between two parents to produce offspring [9]. The first modification to develop PAUX is to change Pu to a variable probability for each crossover action that is calculated using (2):

$$P_u = f_1 / (f_1 + f_2) \quad (2)$$

Where f1 is fitness of the first parent and f2 is second ones. This value is used as the probability of exchanging bits between parents. This mechanism allows parents with higher fitness to contribute more of their genetic makeup to their offspring. Besides, PAUX uses a new concept called Template. Template is used to reflect population statistical

information in behavior of crossover operator. In PAUX, Template is a generated chromosome which is constructed using the entire population with respect to fitness and genetic distribution of population. This generation is done using the following procedure: in each position of the generated chromosome, the entire population is traced and average fitness between all chromosomes with the same gene at that position is calculated, then the gene with highest average fitness, as winner gene, is inserted in the same position of the generated Template. For example, consider Fig. 1:

Sample Chromosomes for One-Max Function

	Fitness
1010010101	5
0001000010	3
1110000010	4
1111111100	8

Position	0	1	2	3	4	5	6	7	8	9
Avg. Fitness for 0	3	4	3	4.5	4	3.5	4	3.5	6.5	5
Avg. Fitness for 1	5.6	6	5.6	5.5	8	6.5	8	6.5	3.5	5

If coding selects 0 in case of equal Avg. fitness for 0 and 1 then template will be: **1111111100**, otherwise template will be **11111111101.**

Fig. 1 Template Generating in PAUX

After generating Template, crossover action begins. In PAUX, parents and Template are involved in the crossover procedure and offspring are generated using the following method: after selecting two parents for crossover and finding suitable crossover probability using (1), both genes at the first position in both parents are compared with each other. If they were not the same, a constant value called Padd is added to probability of selecting the offspring's first gene form the parent that its first gene agrees with the first gene of Template. If both first genes of parents agreed with each other and were not equal to the first gene of Template, then with probability of Pth the first gene of offspring comes from Template. Moreover, if both first genes of the parents were the same and agreed with the first gene of Template, then the first gene of offspring will be the first gene of Template. For example, consider Fig. 2:

Template	1	1	1	1	1	1	1	1	0	0	Fitness
First parent	0	1	0	0	0	1	1	1	0	0	4
Second parent	1	1	1	1	1	0	0	1	0	0	6
Probability of selecting bit from first parent	.1	1	.1	.1	.1	.7	.7	1	1	1	

These values are calculated using these parameters: P_u=4/(4+6)=.4,P_{add}=.3 and P_{th}=.7

Fig. 2 Swapping Probability Calculation in PAUX

Using this method, statistical information of population is also involved to produce new generations. This involvement can help us to improve GA performance as shows in [1]. In addition, with respect to this fact that bit selection is biased using Avg. fitness of chromosomes with the same gene, premature convergence can be avoided or recognized rapidly. This is because only one chromosome with different gene structure and higher fitness than the local minima can produce many offspring similar to it and can change Template in to produce more similar offspring.

V. Statistical Genetic Algorithm (SGA)

In this section, we are going to provide detailed description to SGA's structure. To describe a genetic algorithm operator we must describe these subjects: chromosome representation method, selection operator, crossover operator, mutation operator, and fitness calculation method and operator applications.

Chromosome Representation Method: Chromosome representation method in *SGA* is similar to Simple *GA* and is binary representation. Further information about this representation can be found in [9].

Involved Operators: *SGA* uses pattern based operators: *SSO* as selection operator and *PAUX* as crossover operator. Due to the nature of *PAUX* which is described in previous sections that can produce new offspring with an external gene that could be found in none of its parents, we can assume that *PAUX* implements and implicit mutation procedure so there is no explicit mutation operator in *SGA*.

Operator's Application: The best way to describe operator's application in *SGA* is to describe a life cycle of this algorithm.

At first *pattern* is generated for *PAUX* as described in previous sections. Then two chromosomes are selected randomly from mating pool and apply *PAUX* to them with the probability of P_c to produce new offspring. This procedure is done till number of produced offspring reaches mating pool's numerosity. Then next generation is selected from the population of parents and offspring using *SSO*. This procedure is repeated till *SGA* solves the problem or reaches the maximum number of iterations.

Another important point is the calculation procedure of P_c. To describe this procedure, at first we describe the aim of using crossover operators in *Genetic Algorithms*. Due to [9], crossover operators are used to propagate genetic material of *good* chromosome through the entire population. So if the entire population seems to be *good* then crossover is not very necessary but in the other hand if population seems to be *bad* the crossover operator is highly recommended to propagate genetic materials in the population. Also, another reason to use crossover operator is to escape from local optima where the entire population are going to be genetically uniform around a wrong chromosome. So crossover operator must be apply more times when the population are going to be genetically uniformed to escape from local optima.

Due to this information, we propose a simple model to calculate crossover probability that is presented using (3):

$$P_c = 1 - \left(\frac{M_f}{F_{Best}} + \frac{M_D}{D_{Best}}\right)/2 \qquad (3)$$

Where M_D is mean of *hamming distance* of all chromosomes with the best one (with respect to fitness), M_F is mean of fitness, F_{Best} is best fitness in the population and D_{Worts} is the greatest distance between chromosomes and the best one (using hamming distance).

Fitness Calculation Procedure: Fitness calculation procedure in *SGA* is completely similar to fitness calculation procedure in simple *GA*.

VI. Design of Experiments

In this section, we briefly describe benchmark problems that are used in this paper and test procedure. This problems are chosen with respect to the proposed criteria in [10] and [11] from three families of *simple* and *hard* problems. These problems are described here:

One Max Problem: This problem is one of the well-known GA problems, for a binary string x of length l, this is the problem to maximizing below equation:

$$\sum_{i=1}^{l} x_i, x_i \in \{0,1\} \quad (4)$$

One-Max is a classical test to confirm that one is able to artificially evolve to a solution starting from a given initial population.

Modified Royal Road Problem: The `Royal Road' function(s). In this problem each chromosome is regarded as being composed o regularly-spaced non-overlapping blocks of bits, separated by a number of irrelevant bits. Various parameters control the scoring: b, g, and m* are integers and u*, u and v are real numbers. The low-level blocks are of size b bits, with g irrelevant bits making up a gap between each. Each low-level block scores 0 if completely filled or *mv* if it contains m bits and m -‹ m*, or *–mv* if there are more than m* bits set but the whole block is not filled. Thus the low-level blocks are mildly deceptive. In addition there is a hierarchy of completed blocks which earn bonus points. The hierarchy has a number of levels; level 1 is the lowest, and level (j + 1) has half the number of blocks that level j has - the first and second blocks in level j form the first block of level j + 1, the third and fourth from level j form the second and level j +1 and so on. If level j has nj > 0 'filled' blocks then it earns a bonus score of u* + (nj - 1) u; thus u* is a special bonus for getting at least one filled at that level. The topmost layer of the hierarchy has one block, which is filled if and only if every block in the lowest level is filled. [11] reports that such 'royal road' functions are very hard for some GAs and analyze why. See also the challenge issued by John Holland in the GA list, vol. 7 no. 22. This challenge is used in our experiment; the pseudo code is given in the following:

Royal Road (JH):
j indexes levels in hierarchy (1 is lowest level).
i indexes target schemata (1 is at left).
*There are 2**k target schemata at level 1, and 2**(k-j) target schemata at level j+1 (compounded of adjacent pairs of schemata from the next lower level); each target schema is defined over b loci.*
BONUS(j) = u+(n(j)-1)u, n(j)>0*
= 0, n(j) = 0
where n(j) is the number of found targets at level j and u and u are parameters, u*>u.*
*PART(i)=contribution to overall score from m(i) correct alleles in target schema i at the lowest level, 0<i<1+2**k,*
= m(i)v, if m(i)<m+1,*
= -(m(i)-m)v if m*<m(i)<b,*
= 0 otherwise.
(PART introduces simple nonlinearities: The score actually decreases if there are more than m correct bits in the target area).*
`SCORE = Sumj[BONUS(j)]+Sumi[PART(i)].`

Test experiments are done as follows: at first, entire population is initiated randomly, then algorithm is executed for maximum number of 5,000 iteration for the first problem and 10,000 for the second one. In the end of all iterations, mean fitness is calculated as the performance measure of the algorithm. For both two problems, *SGA*, Simple *GA*, and Simple *GA* with *PAUX* and *SSO* are tested in 25 independent experiments and mean of these 25 runs are plotted as performance of each algorithm.

Used parameters are initiated as follows, population size is set to 50 for first problem and 800 for second one, P_c is set to 0.5 for static rate of crossover, w_1 is set to .4 and w_2 is set to .6 for *SSO* and P_{add} is set to 0.3 and P_{th} is set to 0.7 for *PAUX*. RR parameters are set as follows: *Chromosome Length*=246, b=8, g=7, m^*=4 and u^*=1.0. Results are shown and discussed in the proceeding sections.

VII. Experimental Results

Fig. 3 and 4 indicates mean fitness of population for Simple *GA*, Simple *GA* with *PAUX*, and Simple *GA* with *SSO*, Simple *GA* with *SSO* and *PAUX* and *SGA* for *on- max* and *Modified RR* problems.

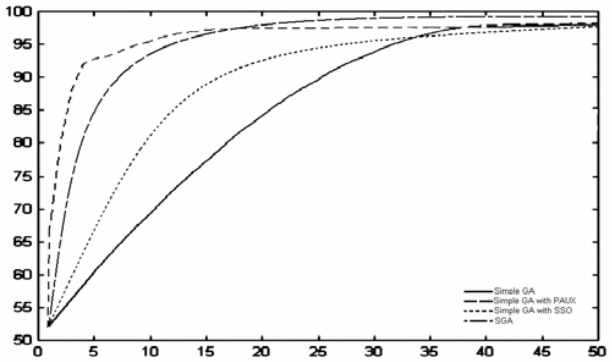

Fig. 3 Comparing SGA, Simple GA, PAUX and SSO in One-Max Problem

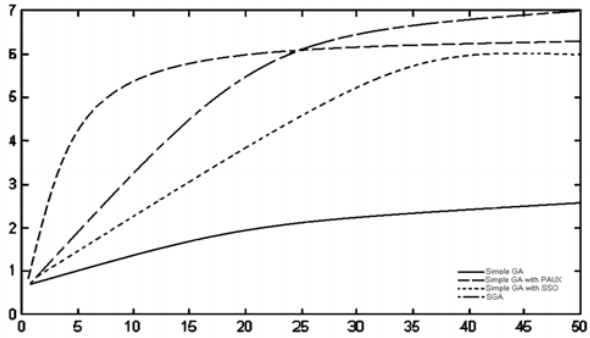

Fig. 4 Comparing SGA, Simple GA, PAUX and SSO in RR Problem

VIII. Discussion

As shown in the experimental results in the previous section, it is very clear that *SGA* works significantly better than simple *GA*. It can be for several reasons such as using *PAUX* and *SSO* that both of them can improve GA performance clearly [2, 3]. However, the interesting point is that *SGA*'s performance is better than simple GA with *PAUX* and simple *GA* with *SSO*. We can propose two reasons: simultaneously usage of *PAUX* and *SSO* and dynamic calculation method of P_c. this hypothesis is tested and we find out that combining *PAUX* and *SSO* can improves performance of simple GA that uses each of them separately, but it cannot reach performance of *SGA* (results are not presented in this paper). We investigate that *SGA* with static rate of P_c has lower performance that *SGA* with dynamic rate. To illustrate this issue consider Fig. 5.

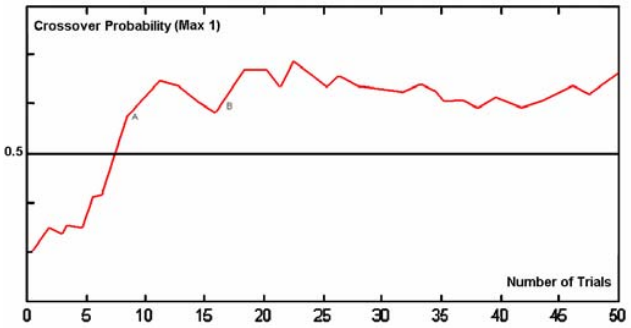

Fig. 5 Crossover probability of SGA in a single run of RR

This Figure indicates P_c during a single run of *SGA* in *RR* problem. With respect to this

Fig. 5 and Fig. 6, that indicates mean distance of the population with the best one (using hamming distance as measurement tool), and Fig. 4 that shows mean fitness of the population for this problem; we can conclude that P_c can help to improve *SGA*'s performance in some states that *PAUX* and *SSO* can not help it. For example, consider points *A* and *B* that are highlights in Fig. 6. These points indicate two critical points that mean fitness of population is near the best fitness (Phenotypic uniqueness) and population diversity decreases slowly. Our proposed mechanism to calculate P_c can overcome this situation by increasing probability of crossover and can produce some new genetic materials to increase population diversity (as shown in Fig. 4, points *C* and *D)*. This issue can help *SGA* to escape local optima in problem landscape better than simple *GA* even with *PAUX* and *SSO*.

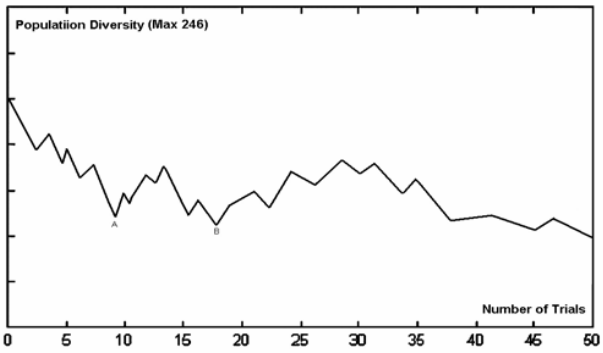

Fig. 6 Population Diversity (Using Hamming Distance) in of SGA in a single run of RR

IX. Summary

In this paper, we employ our previously invented operators such as PAUX [1] and SSO [2] to introduce a new approach to adaptive genetic algorithms that is called Statistical Genetic Algorithm. This approach uses statistical information of the population as a guideline to improve its performance when it applies evolutionary operators to the population. This algorithm is described in detail and compare with traditional GA with some benchmark problems.

References

[1] A. Hamzeh, A. Rahmani (2004), *Adaptive Crossover in Genetic Algorithms using Pattern Based Method*. In proceeding of 9'Th Computer Society of Iran Computer Conference.

[2] A. Hamzeh, A. Rahmani (2005), A New Selection Method for Genetic Algorithms based on Genotypic Information of the Population. In proceedings of 10'The Computer Society of Iran Computer Conference.

[3] S. Yang (2002), *Adaptive Crossover in Genetic Algorithms Using Statistics Mechanism*. In Artificial Life VIII, Standish, Abbass, Bedau (eds)(MIT Press) 2002. pp 182-185.

[4] H. Luchian, O. Gheorghies (2003), *Integrated-Adaptive Genetic Algorithms*, In Proceeding of 7th European Conference on Artificial Life (ECAL 2003), Dortmund, Germany, September 14-17, 2003.

[5] J. Gómez, D. Dasgupta, F.A. González (2003), *Using Adaptive Operators in Genetic Search*. In Proceeding of GECCO 2003: 1580-1581.

[6] J. Gomez, D. Dasgupta (2002), *Using Competitive Operators and a Local Selection Scheme in Genetic Search*. In *Late-breaking papers GECCO 2002*, 2002.

[7] F. Herrera, M. Lozano (2003), *Fuzzy adaptive genetic algorithms: design, taxonomy, and future directions,* In Journal of Soft Computing 7 (2003) 545–562, Springer-Verlag 2003.

[8] Y. Maeda, Q. Li (2005), *Parallel Genetic Algorithm with Adaptive Genetic Parameters Tuned by Fuzzy Reasoning*, International Journal of Innovative Computing, Information and Control Volume 1, Number 1, March 2005 pp 95-107.

[9] M. Mitchell (1996). *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, Massachusetts.

[10] S. Forrest, M. Mitchell (1993), *What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation,* In Machine Learning Journal, Volume 13, Issue 2-3 Nov. /Dec. Special issue on genetic algorithms pp: 285-319.

[11] T.Jones, S.Forrest (1995), *Fitness distance correlation as a measure of problem difficulty for genetic algorithms*. In Larry Eshelman, editor, Proceedings of the Sixth International Conference on Genetic Algorithms, pages 184-192, San Francisco, CA.