Sindh Univ. Res. Jour. (Sci. Ser.) Vol.44 (2) 291-294 (2012)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

An Adaptive Mutation Operator for Global Optimization Problems

I.A. KOREJO, K.BROHI, M.S.VIGHIO*

Institute of Mathematics and Computer Science, University of Sindh, Jamshoro.
Corresponding author: I.A. KOREJO, Email: cskorejo@gmail.com Cell.No. +92-3323605603

Received 17th February 2012and Revised 2nd April 2012)

Abstract: Genetic algorithms are powerful search methods for global optimization problems. Different mutation operators have been suggested in GAs to produce the next generation; it is difficult to choose which mutation operator should be applied in the evolutionary process of GAs. The adaptive mutation approach integrating various mutation operators into a simple GAs; most suitable mutation operator is adaptively selected based on the probability matching technique while solving the problem. In this novel algorithm, each individual has a group of learning strategy along with different behaviour in the search phase. This paper evaluates the performance of our suggested scheme on different problem settings. The experimental result shows that our suggested mechanism is able to select automatically the appropriate mutation operator for different problems. Several mutation operators can obviously enhance the performance of GAs.

Keywords: Genetic Algorithms, Probability Matching and Global optimization problems.

1. INTRODUCTION

Genetic Algorithms (GAs) are stochastic search and optimization methods inspired by principles of natural evolution of species, which were first proposed by John Holland in 1960s (Holland 1975). Nowadays, GAs have been found extremely useful in finding better candidate solutions for numerical global optimization problems. Global optimization problems have arisen in various fields like science, engineering, and business. However, many of these problems are difficult to be solved by the analytical process. GAs have been successfully applied as an efficient technique for solving various optimization problems in different fields.

Several operations are involved for generating next population in genetic algorithms (GA), with each individual in the population representing possible solution. There are basic operations in GA like crossover (the offspring are generated from selected individuals from population, by swapping some genetic information in the two individuals. Hence, offspring inherit some genes from both parents), mutation (mutation operator randomly alters gene values during the evolutionary process. offspring may be different from parent.), selection (selection scheme chooses better individuals for the next generation according to the predefined rules.)

Parameter setting has become one of the most active research directions in GAs. Generally speaking, there are two major classes of methods to set parameter values: parameter tuning and parameter control. Parameter tuning sets parameters before the execution of an algorithm but requires extensive experiments while parameter control adjusts parameters during the running process of an algorithm within reasonable time.

In this paper, we propose adaptive technique, in which more than one mutation operators are used. This adaptive approach selects one of the four mutation operators according to the probability matching technique. Different mutation operators may be able to enhance the performance of GAs, instead of a single mutation operator.

The remaining part of this paper is organised as follows: section II explain the adaptation of mutation in GAs. In section III, we discuss the Design Adaptive Approach for GAs Using Four Mutation Operators, while in section IV the results are discussed, and finally section V concludes the paper.

Adapting the mutation in GAs

Adaptation has become one of the most active research directions in Genetic Algorithms (GAs). The value of algorithm parameters and genetic operators is adjusted within GAs to enhance GAs performance. The concept of adaptation of relevant parameters and genetic operators was first suggested into evolution strategies where mutation step size is adjusted through the self-adaptation (Schwefel, 1998).

There are two main classes of methods to set parameter values: parameter tuning and parameter control (Eiben 2007, Eiben 1999). The parameter tuning is done before the execution of evolutionary process; these values will be applied to assess the algorithm, according to the experience or by trail-and-error approach. However, the EAs researchers eventually discourage static values because different values of parameters and different operators may be suitable at different level of main run of EAs. In order to address

*Department of Information Technology, QUEST, Nawabshah.

this deficiency, researchers have diverted attention towards adapting the parameters during the evolutionary process for finding better solutions to the problem in hand.

Parameter control can be divided in different ways. Deterministic mechanism adjusts the values of parameters according to some predefine rules. Adaptive mechanism modifies the strategy parameters using the feedback information from the search space (Davis 1991,Julstrom 1995). In the case of self-adaptive mechanism encode the parameters within the genotype, which is thus developed in parallel with the individual (Back 1992).

C. *Design Adaptive Approach for GAs Using Four Mutation Operators*

Single mutation operator is applied to produce offspring in the conventional genetic algorithms. More than one mutation operators can be used to help GAs jump out of local optima. However, the effectiveness of these mutation operators varies on different levels of optimization problems, even same on the specific problem. The different mutation operator can achieve best performance at different stages of the GAs, instead of single mutation operator. This paper suggests adaptive mutation operator that can adaptively select most suitable mutation operator according to the probability matching approach for different problems. Before showing an adaptive mutation operator, four mutation operators proposed in GA's literature(Hong 2000).

A. *Learning pool*

Different mutation techniques have been introduced for GAs (Hong 2000). These mutation strategies and/or recombination operators are employing in the reproduction level. The learning pool uses four mutation operators, denoted M1 to M4, respectively. The detail description of these mutation operators are given in (Hong 2000).

B. *Probability Matching Scheme for Allocating Operator Probabilities (PMS-AOP)*

The general approach of Probability Matching (PM) is defined by (A, P, R, Q). Suppose, we have a set of K operators $A = \{a_1, ... a_k\}$, and probability vector

$$P(t) = \{p_1(t), ...p_k(t)\} (\forall (t) : 0 \le p_i(t) \le 1; \sum_{i=1}^{k} p_i(t) = 1)$$

The probability $p_a(t)$ of each operator a is update according to its known performance (frequently updated by the rewards received) by using the PM mechanism. The reward value $R_a(t)$ of operator a at generation t is received after its performance. In addition, $Q_a(t)$ is the quality vector that stipulates a running estimate of the reward for each operator. Quality vector is altered as follows:

$$Q_a(t+1) = Q_a(t) + \alpha[R_a(t) - Q_a(t) \quad (1)$$

With the adaptation rate $\alpha \in (0,1]$.

$$p_a(t+1) = P_{\min} + (1 - K.P_{\min}) \frac{Q_a(t)}{\sum_{i=1}^{k} Q_i(t)} \quad (2)$$

where $P_{\min}$ denotes the minimum value of each mutation operator, applied to ensure that no operator get lost (Thierens 2007).

Assigning a credit value to mutation operators

In order to allocate a credit value to an operator based on the fitness improvements of the solutions affected by that operator (Hong 2000, Li 2008). The mean value is calculated of each operator's credit value. Several approaches can be used to calculate the mean value of the fitness improvements; for example simple average, winsorized mean, and weighted mean.

GA with Adaptive approach

Probability Matching Based Adaptive Mutation Operator (PM-AMOGA) is developed with the help of above mentioned three aspects (Learning pool, PMS-AOP, and Credit assignment) for the GA algorithm. The framework of PM-AMOGA is shown in algorithm 2. Some steps are modified in the basic GA algorithm and these steps are marked with right arrow. At each generation t, each mutation operator is used based on its probability and its offspring fitness is evaluated. After that, the relative fitness improvement is calculated and stored. Finally, the reward, quality and probability of each mutation operator is updated at the end of each generation.

Algorithm 1 Probability matching based adaptive mutation operator for GA optimizer

Randomly generate initial population
Evaluate the fitness of each individual of population
t := 0;
Set K := 4 and $P_{\min}$:= 0. 05 ←
For each operator a , set $Q_a(t)$:= 0 and $p_a(t)$:= 1/ K ←
while t < max_gen **do**
for i := 0 to popsize **do**
Select individual j by the roulette wheel mechanism
Crossover solution i with solution j using uniform crossover method
Choose the mutation operator a for i
individual based on its probability ←
Mutate the solution i by applying selected mutation operator a
end for
for i := 0 to popsize **do**
Evaluate the offspring c_i

if $f(c_i)$ is higher than or equal to $f(p_i)$ **then** ←
Calculate progress value from (Li 2008) ←
end if
end for
Calculate the reward $R_a(t)$ for each mutation operator ←
Update the quality vector $Q_a(t)$ for each mutation operator ←
Update the probability vector $p_a(t)$ for each mutation operator ←
$t := t + 1$;
end while

4. EXPERIMENTS AND RESULTS

A. Test Functions

In order to investigate the performance of PM_AMOGA operator with compared algorithms on different multimodality test-problem generator proposed by Spear 2000, which are recently applied as a test-problem generators in the literature Vafaee 2010. Generally, these test bids can generate random problems from certain classes of problems with user-controllable properties. We choose multi-modal functions to examine how well our proposed approach can explore the landscape and exploit true optima as the number of peaks in the search space varies.

B. PM_AMOGA mutation operator Compared with Other mutation operators.

To compare the performance of proposed PM_AMOGA with traditional mutation operators such as M1 to M4, which are mentioned in the section III of this paper. Why we choose these mutation operator because these are widely applied on different benchmark methods in the literature.

C. Parameter Setting

In order to evaluate the performance of PM_AMOGA operator with other four mutation operators with respect to the different levels of difficulty, we altered the length of the solutions l and the population size n to vary the difficulty of the problems. Clearly, the problems become more challenging task if l increases. We have studied different levels of problem difficulty by setting the parameter pair (l, n) to (10, 10), (20, 30), and (30, 50), respectively. We applied each of these parameter pair to multi-modal landscape with different degrees of multi-modality by assigning the number of peaks to the values in 1,5,10.

D. Experimental Results and Analysis

Given the randomize behaviour of the GA algorithm, the average result of 50 independent runs of each algorithm on different multi-modality problems. For each run of an algorithm on specific problem, 300 generations were allowed. In the light of the end-of-run result shown in the Table I. It can be observed that the performance of PM_AMOGA operator is different on the identical (l,n)-setting of different problem setting. Single mutation operator may be more effective on some optimization problems and may be worse on other functions. PM_AMOGA operator is the second best operator among the rest of the operators on most problems. All five algorithms with mutation operators perform identical on few problems. PM_AMOGA operator performs worse among the mutation operators on the problem setting of peak 1, l =10, and n =10.

Table I: Average best result achieved by various mutation operators for different problem settings.

Peak number =1				
(l, n)		(10,10)	(20,30)	(30,50)
AM	Avg	0.9966	0.9483	0.8977
M1	Avg	1.0	0.9299	0.8999
M2	Avg	1.0	0.9566	0.8611
M3	Avg	1.0	0.9333	0.8777
M4	Avg	1.0	0.9383	0.8688
Peak number =5				
(l, n)		(10,10)	(20,30)	(30,50)
AM	Avg	1.0	0.9499	0.8666
M1	Avg	1.0	0.9649	0.8600
M2	Avg	1.0	0.9466	0.8911
M3	Avg	1.0	0.9383	0.8988
M4	Avg	1.0	0.9349	0.8755
Peak number =10				
(l, n)		(10,10)	(20,30)	(30,50)
AM	Avg	1.0	0.9433	0.8855
M1	Avg	1.0	0.9716	0.8600
M2	Avg	1.0	0.9366	0.8755
M3	Avg	0.9966	0.9600	0.8644
M4	Avg	1.0	0.9366	0.9133

As expected that the performance of PM_AMOGA is become second or third best mutation operator among all mutation algorithms on different benchmark optimization problems. It can be observed from **(Fig.1)** that the best fitness value of individuals was considered at every iteration for 50 independent runs and the average is plotted with respect to the generation. From **(Fig.1).** It can be shown that the convergence curve of PM_AMOGA is second best faster than other mutation algorithms on different problem settings. This Fig. presents only few graphs of various mutation algorithms.

(**Fig.2)** shows the result of selection ratio of M1to M4 at generation 300 for 50 independent runs in the PM_AMOGA. There is lot of fluctuation during the evolution of the four mutation operators. These mutation algorithms may have their own behaviour during the whole optimization process on all test problems. Due to this reason, different mutation operators can locate different search directions in search phases. It is possible to help this approach to find the global or local optima.

4. CONCLUSIONS

This paper presents a method to adjust the probabilities of each mutation operator according to the

probability matching approach and relative fitness improvement scheme. Which integrates the M1, M2, M3, and M4 operators. The performance of different mutation operators varies on different types of optimization problems, PM_AMOGA operator presents a balanced performance on all benchmark problems and it determines second best result on most of the problems. The core of adaptive mechanism is to automatically adjust genetic operators and relevant parameters in order to speed up the convergence process as well as maintaining the population diversity. When tackling the multiple peaks problems, the maintenance of small amount of population diversity is very useful for the fitness landscape, and thus should also be taken into account for the rewarding of the each mutation operators. Hence, adaptive scheme is extremely useful for improving the performance of GAs.

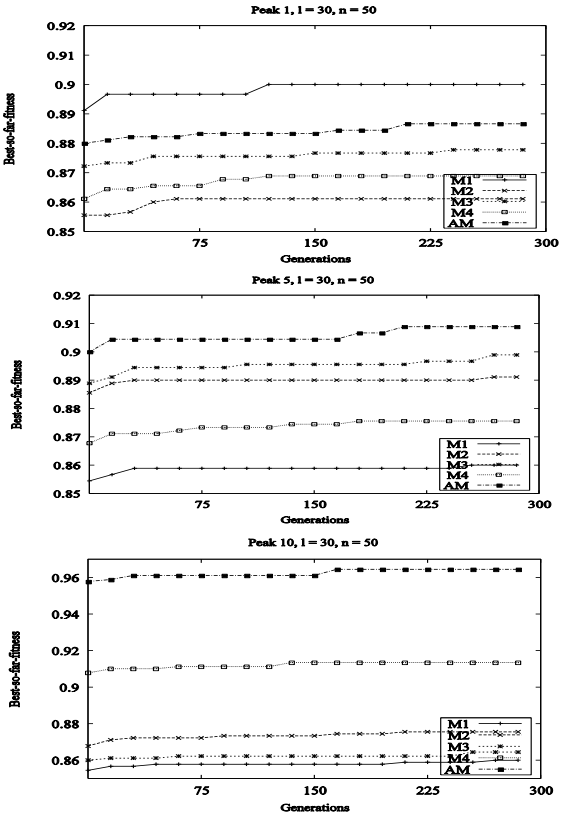
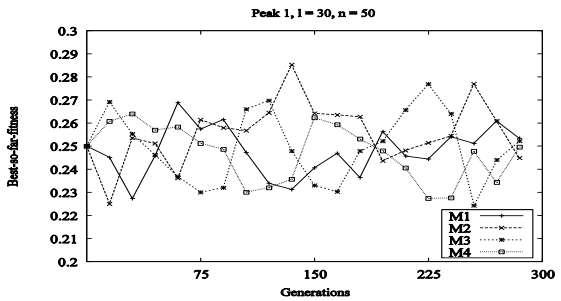


Fig. 1 Evolutionary process of PM_AMOGA, M1, M2, M3, M4 on different parameter setting of different peaks



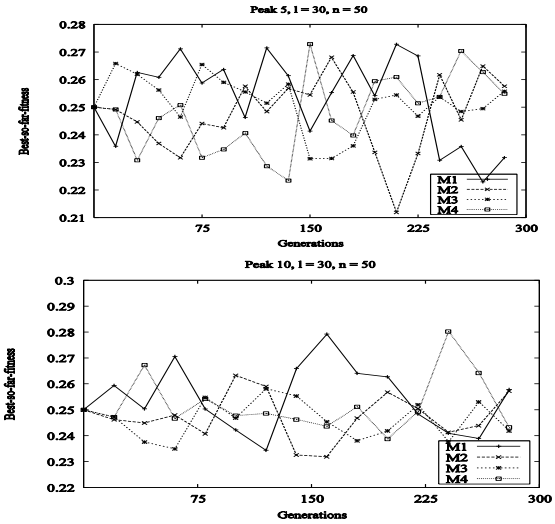


Fig.2: The average selection ratio of PM_AMOGA, M1, M2, M3, M4 on different parameter setting of different peaks.

REFERENCES:

Davis L. (1991) Handbook of Genetic Algorithms. Van Nostrand Reinhold.

Eiben, A. E., Z.Michalewicz, ,M. Schoenauer, and J. E. Smith (2007) Parameter control in evolutionary algorithms. In Lobo, F.G. et al., editor, *Parameter Setting in Evolutionary algorithms*, 19-46. Springer.

Eiben, A.E., R. Hinterding and Z. Michalewicz (1999) Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 3 **(2):** 124–141.

Holland J. H. (1975) Adaptation in natural and artificial systems. Ph.D thesis, Ann Arbor, MI, USA.

Hong, T.P., H.S. Wang, and W.C. Chen (2000) Simultaneously applying multiple mutation operators in genetic algorithms. Journal of Heuristics, **(6):** 439–455.

Li, C., S. Yang, and I. Korejo (2008) An adaptive mutation operator for particle swarm optimization. In Proceedings of the (2008) UK Workshop on Computational Intelligence, 165–170.

Schwefel H.P. (1998) Numerical Optimization of Computer Models . Wiley, Chichester.

Spears W. M. (2000) Evolutionary Algorithms: The Role of Mutation and Recombination. Natural Computing Series. Springer, Secaucus, NJ, USA.

Thierens D. (2007) Adaptive strategies for operator allocation. In F.G. Lobo, C. F. Lima, and Z. Michalewicz, editors, Parameter Setting in Evolutionary Algorithms, of Studies in Computational Intelligence, vol. **(54):** 77–90.

Vafaee F. and P. C. Nelson (2010) An explorative and exploitative mutation scheme. In Proceedings of the IEEE Congress on Evolutionary Computation, 1–8.