Sindh Univ. Res. Jour. (Sci. Ser.) Vol.45 (1): 41- 48 (2013)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Genetic Algorithm Using an Adaptive Mutation Operator for Numerical Optimization Functions

I.A. KOREJO, Z. U. A. KHUHRO++, F. A. JOKHIO*, F. N. MEMON, QURAT-UL-AIN NIZAMANI

Institute of Mathematics and Computer Science, University of Sindh, Jamshoro

17 January 2012 Received and Revised 19 February 2013

Abstract: Simple genetic algorithms (GAs) applies only one mutation operator to produce next population. Four mutation operators have been suggested in GAs. Specifying which mutation operator should be employed is very difficult but not impossible and is normally done by empirical results. In this paper, new adaptive approach is proposed to solve the problem. This scheme uses more than one mutation operator to produce next generation. It modifies the selection mutation ratio of each mutation operator based on the global behaviour of the population for each generation. The main idea of the adaptive algorithm is to alter the operators' probabilities based on the feedback information of population for each generation. This technique is conducted on a fifteen benchmark optimization problems. The result show that the adaptive mutation accomplishes equally well for all of the test functions.

Keywords: Genetic Algorithms, adaptive mutation operator and Global optimization problems.

1. INTRODUCTION

Genetic algorithms (GAs) are based on optimization search method from the principles of natural selection and population genetics. It maintains a population of potential solutions to a given problem; each solution is evaluated according to a problem-specific fitness function that defines the evolution environment for the individuals. New population is generated by choosing best solutions from the current population and applying the genetic operation on it. Original version of GAs was proposed by John Holland (Holland, 1975). Since last few decades, GA's are best known and have been applied for solving many optimization problems for finding the good solutions to numerical global optimization problems.

Parameter setting is the prominent active research area in the field of evolutionary algorithms (EAs) since last few years (Eiben, 1999; Eiben, 2007). Feedback information taken from Design Of Experiments have been modified to the off-line setting of relevant parameters and genetic operator of an EA; while they are more effective than traditional ANOVA. These approaches might be time consuming regarding the experiments. Online tuning is altered on the basis of several mechanisms during the execution of EAs (Davis, 1991; Schwefel, 2007; Vafaee, 2010; Thierens, 2007). This class of online tuning can be further divided into subclasses; interested researches can refer (Eiben, 2007).

In this paper, a new adaptive mutation operator is introduced for GAs. The goal of this approach is to integrate the four mutation operators into one algorithm to achieve better solutions in the search space. The adaptive approach shows a balanced performance on benchmark problems and it provides the best performance on few problems.

The rest of this paper is managed as follows. Section 2 provides a detailed description of an adaptive mutation operator method for GAs. Experimental results are presented in section 3 in order to compare the performance of the analysed mutation operators for GAs on a set of benchmark test problems. Finally, section 4 describes conclusion and brief overview for the future work.

2. MATERIALS AND METHODS

Adaptive approach can be applied to enhance the performance of GAs by dynamically tuning the probabilities of operators according to the population feedback instead of solely relying on a prior knowledge. The main idea of the adaptive mutation technique is to alter the probabilities of operators regarding the global behaviour of the population for each generation. GAs can be able to jump out from local optima due to different mutation operators. It is possible that the performance of mutation operator may vary on different optimization problems or remains unchanged. However, single mutation operator cannot get the best results; instead, several mutation operators may be used at different levels of evolutionary process, for the best performance. This paper focuses an adaptive mutation operator that can automatically choose the most appropriate mutation operator for different optimization problems. The following section would explain four mutation operators used in this paper.

++Corresponding author, Z.U.A. Khuhro, email: kzainkhuhro@yahoo.com Ph. No.+92-333-1123976
*Department of Computer System, Engineering, Quaid-e-Awam University, Nawabshah

2.1. Mutation Operator Pool

Various researchers have proposed schemes in simple GAs (Hong, 2000; Korejo, 2012), applying different mutation and crossover operators in the reproduction level. In order to design adaptive approach used in this paper. Four mutation operators described as follows:

A. Flip bit Mutation

It is conventional mutation operator in the genetic algorithms, which alters the bit values 0 to 1 and 1 to 0 (Hong 2000).

B. SANUM Mutation Operator

Yang in 2004 proposed a statistics-based adaptive non-uniform mutation operator for GAs, which is gene level adaptive mutation operator. Initially, it calculates the number of ones for each locus in the current population and then modifies the mutation probability for that locus during the evolutionary process of GA.

C. GBAM Mutation Operator

Gene based adaptive mutation (GBAM) operator was introduced by Uyar in 2004. This technique uses an asymmetric adaptive mechanism on each locus of the solution. GBAM approach has two different mutation rates for each locus: P_m^1 for those genes that have alleles of "1" and P_m^0 for those genes that have alleles of "0". The probabilities of P_m^1 and P_m^0 are automatically updated based on the global behaviour of the population for each iteration, according to the relative success or failure of individuals having a "1" or "0" at that locus for each generation.

D. GBAM_FAD Mutation Operator

Yang and Uyar in 2005 introduced a gene based adaptive mutation with fitness & allele distribution correlation scheme, which constructs probabilities of each gene locus with the combination of information of fitness and allele distribution. Two different probabilities of each gene locus are used in GBAM_FAD similar as in GBAM. For each iteration, the mutation rates for each gene locus are updated based on global behaviour of the current population, regarding to the relative success or failure of those solutions.

2.2. Design Adaptive Mutation Technique for GAs Using Four Mutation Operators

Several mutation operators have been applied in simple Genetic Algorithms (Gas), each operator have different properties and being suitable for different problem (Hong, 2000; Spear, 2000 Yang, 2003; Uyar, 2004; Yang 2006). However, selecting the best mutation operator for a specific problem at hand is difficult task. The adaptive mutation scheme can achieve best performance at different levels of GAs, instead of single mutation operator.

The four mutation operators are used in the proposed approach as suggested before. These mutation operators have an equal initial selection ratio with 1/4. Each mutation operator was employed to its selection ratio and improvements of all solutions affected by that operator. The selection ratio of each mutation operator will increase if the mutation operators generate the best offspring than their parents. The selection ratio of each mutation operator will decrease if the mutation operator created the worst offspring than their parents. The most appropriate mutation operator will be selected adaptively and balance all the mutation behaviour over the whole population.

Some basic definitions of the proposed algorithm are as follows. The progress value $P_i(t)$ of operator i at generation t is calculated as follows:

$$P_i(t) = \sum_{j=0}^{M_i} f\left(x_j^i(t)\right) - \min\left(f(x_j^i(t)), f(\acute{x}_j^i(t))\right), \quad (1)$$

Where $x_j^i(t)$ and $\acute{x}_j^i(t)$ represents parent and its child respectively generated by mutation operator i at generation t and M_i is the total number of individuals that choose mutation operator to i mutate. The reward value $R_i(t)$ of operator i at generation t is calculated as follows:

$$R_i(t) = exp\left(\frac{P_i(t)}{\sum_{j=1}^{N} P_j(t)}\alpha + \frac{s_i}{M_i}(1-\alpha)\right) + c_i p_i(t) - 1, \quad (2)$$

Where s_i is the set of all relative fitness improvements accomplished by mutation operator i during generation t; $p_i(t)$ is the selection ratio of mutation operator i at generation t, $\alpha \in (0,1)$ is the random number, N is the number of mutation operators, and c_i is the penalty factor for mutation operator i, which is defined as follows:

$$c_i = \begin{cases} 0.9, & if\ s_i = 0\ and\ p_i(t) = max_{j=1}^{N}(p_i(t)) \\ 1, & otherwise \end{cases} \quad (3)$$

If the previous best operator is not involved in whole population than the selection ratio of the current

best operator will decrease. With the above definitions, the selection ratio of mutation operator i at iteration t is adjusted using following equation.

$$P_i(t+1) = \frac{R_i(t)}{\sum_{j=1}^{N} R_j(t)}(1 - N * \gamma) + \gamma \qquad (4)$$

Where γ = 0.01 is the minimum mutation ratio of each mutation operator. This selection ratio modify equation consider four factors i.e reward value, the ratio of successful mutations, previous selection ratio and the minimum mutation selection ratio.

The framework of GA algorithm with one of the four mutation operators described above is presented in algorithm 1. There is one difference between the original GA and GA with adaptive mutation operator at lines 4, 5, 7, and 9. One of the four mutation operators selected according to their selection ratios to mutate solution at t generation on same line and then updates the selection ratio for each mutation operator according to equation 4.

Algorithm 1 GA with adaptive mutation operator:

1. Randomly generate the individuals of Population
2. Evaluate the fitness of each individual
3. Apply the crossover operator to create Children
4. Assign various mutation operators, each with an initial selection mutation ratio
5. Perform the mutation operator to mutate individual (each mutation operator applies its selection mutation ratio)
6. Evaluate the fitness of each individual
7. Calculate the progress value according to the equation 1.
8. Perform the selection approach to choose the individuals according to their fitness values.
9. Modify the selection mutation ratio of each mutation operator according to the equation 4.
10. If the termination condition is not satisfied, go to step 4; otherwise stop the algorithms.

3. RESULTS AND DISCUSSION

In order to measure the performance of proposed algorithm, 15 benchmark optimization problems $f_{01} - f_{15}$ were selected from the test suit (Yao, 1999). Researchers have widely used these problems in these experiments (Dong, 2007; Li, 2008). Functions $f_{01} - f_{05}$ are uni-modal functions, their names are Sphere, Rosenbrock, Schwefel_1_2, Schwefel_2_21,and Schwefel_2_2.2 functions. Function f_{06} is the step function and its behaviour is discontinuous and one minimum. Function f_{07} is a noisy quartic function along with $random\ (0,1)$. The remaining problems are multi-modal problems where the number of local minimum increases exponentially with the problem dimension. The detailed description of these functions is given in Table 1.

The general parameter setting is as follows: 30, 100 and 300 for length, population size, and maximum iterations respectively. Probabilities of crossover and mutation were set as suggested by (korejo, 2012). The initial selection ratio was 1/4 for each adaptive mutation operator and the minimum selection ratio was set to 0.001 for each adaptive mutation operator. For GBAM and GBAM _FAD, the following parameters were used :=γ 0.001, $[P_{min}, P_{max}] = [0.0001, 0.2]$, and initially $pm^1(i,0) = pm^0 = (i,0) = 0.01$ for each gene i. The values of α (0.05), β (0.04), and P_{min}(0.0001) were fixed for SANUM operator. Tournament selection with the tournament size 2, elitism size 1, and 2 point crossover were allowed to generate next population.

3.1. Results and their analysis

According to the stochastic behaviour of Genetic Algorithms (GAs), all experiments have been repeated 30 independent runs of each algorithm on every test function.

The average best fitness result of experiments shown in the associated Table 2 and graphs pointed out that the proposed approach competes to the other mutation operators.

The experimental results given in Table 2 show that different mutation operator are optimal at different benchmark optimization problems during the evolutionary process. GBAM_FAD operator is effective on all optimization benchmark functions except f_2, f_4, f_{10}, f_{13}, and f_{15}. First three mutation operators are equal due to performance on function f_6. On function f_2, the performance of mutation operators is ordered in such sequence: GBAM, GBAM_FAD, BitFlip, AdaptMut and SANUM. For function f_4, GBAM is better than rest of the algorithms.

Table 1: The detail description of benchmark functions, where n is the number of variables of a function, D $\in R_n$, f_{min} is the minimum value of function.

Test problems	n	D	f_{min}
$f_1(x)=\sum_{i=1}^{n} x_i^2$	10	[-100,100]	0
$f_2(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	10	[-5.12,5.12]	0
$f_3=\sum_{i=1}^{n}\left(\sum_{k=0}^{k_{max}}\left[a^k\cos\left(2\pi b^k(x_i+0.5)\right)\right]\right)-n\sum_{k=0}^{k_{max}}[a^k\cos(\pi b^k)], a=0.5, b=3, k_{max}=20$	10	[-0.5,0.5]	0
$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}(x_i-100)2-\prod_{i=1}^{n}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)+1$	10	[-600, 600]	0
$f_5(x)=-20\,exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-exp\left(\frac{1}{2}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	10	[-32, 32]	0
$f_6(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)2$	10	[-100,100]	0
$f_7(x)=\sum_{i=1}^{n} ix_i^4+\cup(0,1)$	10	[-1.28, 1.28]	0
$f_8(x)=\sum_{i=1}^{n}100(x_{i+1}^2-x_i)2+(x_i-1)2$	10	[-30, 30]	0
$f_9(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	10	[-500, 500]	-4189
$f_{10}(x)=418.9829.n+\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	10	[-500, 500]	0
$f_{11}(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	10	[-10, 10]	0
$f_{12}(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)2$	10	[-100,100]	0
$f_{13}(x)=max_{i=1}^{n}\lvert x_i\rvert$	10	[-100, 100]	0
$f_{14}(x)=\frac{\pi}{30}\left\{10\,sin^2(y_1-1)+\sum_{i=1}^{n-1}(y_i-1)^2.[1+10sin^2(y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}\mu(x_i,5,100,4), y_i=1+(x_i+1)/4$	10	[-50, 50]	0
$f_{15}(x)=0.1\left\{10sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2.[1+sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}\mu(x_i,5,100,4)$	10	[-50, 50]	0

Table 2: Different mutation operators are optimal at different benchmark optimization problems

Problems	AdpatMut	GBAM	GBAM_FAD	BitFlip	SANUM
f_1	0.638712	0.134315	0.11971	808.944	534.591
f_2	0.400526	0.00790084	0.107351	0.236632	25.9933
f_3	0.30488	0.0628645	0.0516898	0.334016	5.55939
f_4	0.454202	**0.320256**	0.359815	0.414808	4.69079
f_5	0.348799	0.430515	0.120471	0.459621	8.86403
f_6	0	0	0	798.7	514.633
f_7	0.00732685	0.00330299	0.00280377	0.0741859	0.369754
f_8	0.0115498	0.0176247	0.00809685	0.0109021	0.0196199
f_9	6.11001	5.05127	4.72568	14.8601	50.4304
f_{10}	-4189.14	-4189.69	-4189.76	-4189.12	-3389.49
f_{11}	0.493709	0.034692	0.0261807	0.821158	911.232
f_{12}	0.0944277	0.0132328	0.012714	0.0963722	4.85074
f_{13}	15.1203	17.1536	31.4859	16.7632	4672.04
f_{14}	1.16506	0.712235	0.651832	1.23177	35.6585
f_{15}	0.183874	0.969754	1.24035	0.156322	516626

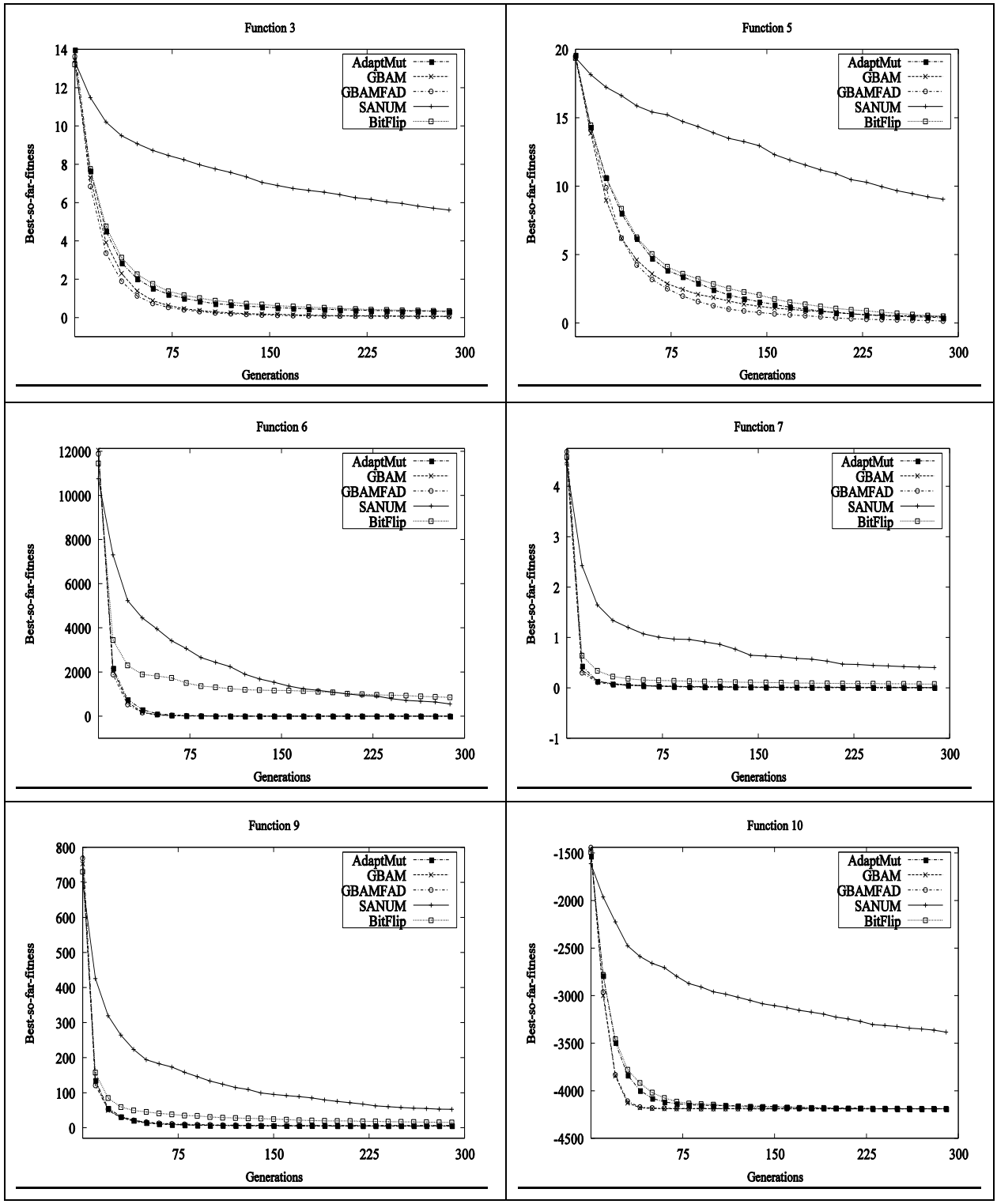
Specially, the result of AdaptMut is better than the other four algorithms on function f_{13}. On function f_{15}, the efficiency of AdaptMut is the second best among the other algorithms. AdaptMut gets closer results to GBAM_FAD and GBAM on most of the test functions.

BitFlip is more effective than the other four mutation operators on function f_{15}. **(Fig.1)** shows the convergence speed exposed by the suggested different evolutionary processes on the various test optimization functions. The best fitness value of the population was recorded at every iteration during the 30 independent runs and the average best fitness is plotted against the generation.

According to the result in **(Table 2 and Fig. 1),** it can be seen that the fitness convergence curve of AdaptMut, GBAM_FAD, and GBAM is almost similar on most of the functions. There is one motivation to see that different mutation operators have different convergence speeds on different functions.

In summary, from the above analysis we conclude that the performance of five mutation operators is reasonably fine except SANUM. The proposed mutation algorithm is better than the other mutation algorithms on few problems.

Fig.1: The fitness convergence curves of AdaptMut, GBAM, GBAM_FAD, BitFlip and SANUM on selected functions.



4. CONCLUSION

A new genetic algorithm, the adaptive mutation genetic algorithm has been proposed for automatically selecting appropriate mutation operators during the evolutionary process for solving the optimization problems. The proposed algorithm integrates the, GBAM_FAD, GBAM, BitFlip and SANUM operators. However, different mutation operators give a different performance on various test

problems, the adaptive mutation operator illustrates a balanced performance on all benchmark functions and it provides the similar performance on few problems. The experimental results show the robustness of the proposed algorithm that is better than other mutation algorithms on few test functions. Our algorithm is very suitable for functions where the appropriate mutation operators are unknown.

REFERENCES:

Davis L. (1991) Handbook of Genetic Algorithms, *Van Nostrand Reinhold New York*, 115.

Dong, H. H., J. H. Huang, and W. Hou, (2007) Evolutionary Programming Using A Mixed Mutation Strategy, *Journal of information science*, 177, 312-327.

Eiben, A. E., Z. Michalewicz, M. Schoenauer and J. Smith, (2007) Parameter control in evolutionary algorithms, *Parameter Setting in Evolutionary algorithms, Springer-Verlag, Berlin, Heidelberg*, 54 (54), 19-46.

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

Holland J. H., (1975) Adaptation in natural and artificial systems, *Ann Arbor, MI, University of Michigan press*, 1 (97), 5Pp.

Hong, T. P., H. S. Wang, and W. C. Chen, (2000) Simultaneously applying multiple mutation operators in genetic algorithms, *Journal of Heuristics, Springer*, 6 (4), 439–455.

Korejo, I. A, K. Brohi, and M. S. Vighio, (2012) An adaptive mutation operator for global optimization problems, *Sindh University Research Journal (Science Series)*, 44 (2), 291-294.

Li, C., S. Yang, and I. A. Korejo, (2008) An adaptive mutation operator for particle swarm optimization, *The 2008 UK Workshop on Computational Intelligence, MIC2008*, 165–170.

Schwefel H.P. (2007) Numerical Optimization of Computer Models, (M. Finnis, Trans.), *John Wiley*.

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

Thierens, D., (2007) Adaptive strategies for operator allocation, *Parameter setting in evolutionary algorithms, Springer-Verlag, Berlin, Heidelberg*, 77-90.

Uyar, S., S. Sariel, and G. Eryigit, (2004) A gene based adaptive mutation strategy for genetic algorithms. *Proceedings of Genetic and Evolutionary Computation--GECCO 2004, Springer-Verlag, Berlin, Heidelberg*, 271-281.

Vafaee, F., and P. C. Nelson, (2010) An explorative and exploitative mutation scheme, *Proceedings of evolutionary computation (CEC), 2010 IEEE Congress on*, IEEE, 1–8.

Yang, S., (2004) Adaptive mutation using statistics mechanism for genetic algorithms, *Research and Development in Intelligent Systems XX, Brunel University Research Archive (BURA,) Springer-Verlag*, 19-32.

Yang, S., and S. Uyar, (2006) Adaptive mutation with fitness and allele distribution correlation for genetic algorithms, *Proceedings of the 2006 ACM symposium on Applied computing, ACM*, 940-944.

Yao, X., Y. Liu, and G. Lin, (1999) Evolutionary programming made faster, *Evolutionary Computation, IEEE Transactions on, IEEE*, 3 (2), 82-102.