# IMPROVING PERFORMANCE OF GENETIC ALGORITHMS USING DIVERSE OFFSPRING AND DYNAMIC MUTATION RATE

R.G. S A. Perera[1] and Lanka Udawatta[2]

[1] Department of Measurements Units, Standard and Services, Sri Lanka
Email: rgsanjaya@yahoo.com
[2] Professor of Electrical Engineering, Department of Electrical Engineering, University of Moratuwa
Email: lanka@elect.mrt.lk

**ABSTRACT**

A Genetic Algorithm coding and a required genetic operation library has been developed with some modifications by introducing dynamic mutation rates and fraction of diverse offspring to improve the searching probability. Improvement was done to the algorithm to automatically select the dynamic mutation rate and fraction of diverse offspring depending on the optimization problem. Modified genetic algorithm with dynamic mutation and diverse offspring was tested with Sin, Step, Sphere and Rasterign benchmark functions and same benchmark test was done with simple random search and conventional genetic algorithm to compare the performance. Also these results were compared with other researchers' results. Results show that the genetic algorithm with Dynamic Mutation rates and Diverse Offspring has better searching performance than the conventional genetic algorithm and the simple random work especially with high dimensional benchmark functions. It also shows that the risk of convergence to local optima can be reduce by the introduction of diverse offspring to the subsequent generations. It shows that the searching performance of a genetic algorithm can be significantly improved by increasing the diversity of the population using dynamic mutation rates and appropriate fraction of diverse offspring while conserving the convergence characteristics. Results show the effectiveness of the proposed algorithm.

***Key words*:** Genetic Algorithm, Diverse Offspring, Dynamic Mutation

## 1. INTRODUCTION

Genetic algorithm is a global search method which is employed the rules of natural evaluation to find the optimized solutions. This was initially proposed by "Holland"[7]. Genetic algorithms are operated on parallel search mechanism to search the optimum result not like other serial mechanisms commonly found on computer based searching algorithms. Latest research results shows that as a global search method genetic algorithms shows superior performance with complex optimizations with large number of decision variables where the other algorithms and methods fails. Many research conduct to improve the performance of genetic algorithms further by introducing different form of genetic operations as well as structural modifications. Multipopulation[6], self adaptive mutation[4], hybrid genetic algorithms[2], and Mimitic algorithms[3] are some of such improvements. Many demanding applications in Artificial Intelligence, Meteorology and other areas like real world manufacturing environments[11] found where optimizations with large and extremely large data sets with complex and hidden patterns has to be analyzed. While the genetic algorithm based on natural evolution mechanisms, performance of genetic algorithms are mainly depend on the implementation of these mechanisms such as crossover, mutation, selection in genetic algorithm coding. Hence fine tune these genetic mechanisms improve the convergence characteristics and searching ability which extend the application area of genetic algorithms in real world problems.

## 2. METHODOLOGY

Genetic algorithm operates by converging individuals towards the optimum eventually. At the beginning individual values in the population is totally random in nature. After generation by generation the crossover operator biases the population towards the optimum. Mutation operator improves the diversity of the population with destruction of some amount of information about the optimum from the population. High bias by crossover improves the convergence rate. But it increases the risk of convergence to a local optimum. High mutation rate reduce the risk of false convergence. But it reduces the convergence rate. Hence there is an optimum balance between the convergence rate and searching ability. Here the method focuses on the mutation. Because if high bias induces by the crossover and high mutation rate apply in a way that least information is lost by the mutation, performance of the genetic algorithm will be improved. For such mutation two methods were considered.

1. Dynamic mutation rates with high mutation rates with worst individuals

2. Insertion of diverse offspring in subsequent generations.

When the individual value is near the optimum low mutation rates increases the rate of convergence. This behavior can be implemented as in figure 2.1, where mutation applied depending on the fitness value of individuals. This profile ensures the best individuals are not altered by the mutation. Also it expand the potential of changing the worst individuals to there values to research to optimum by high mutation. In this way the worst individuals has an opportunity to occupy large span of the search space which improve the searching ability while keeping the minimum information lost by the mutation high mutation only with worst individuals.

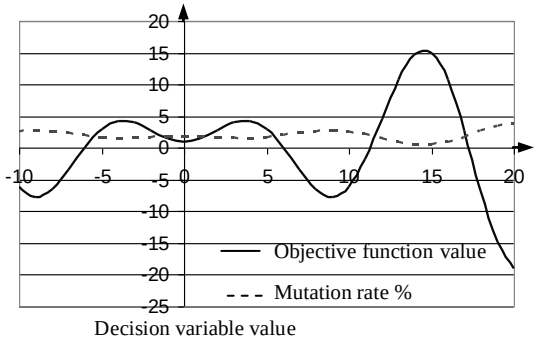


Figure 2.1: Illastration of dynamic mutation rate varies with tha span of search space.

Insertion of diverse offspring (Worst valued individuals) to subsequent genarations will improve the searching ability by the above proposed mutatiom profile. Better offspring near a optimum always evolve towards an optimum by genatic operations(See Figure 2.2-Individual B). In this case mutation can not easily change its direction of convergance, because it applies very small probability and in small span(See figure 2.1). Hence, if an individual is alredy near a local optimum, it will most probably lock in to this local optimum. But diverse offspring are at the bottom of an optimum (See Figure 2.2-Individual C), therefore mutation and crossover of these diverse offspring can produce next genaration not only derected to this local optimum but also in other directions aswell. Above behavior gives ability to find the global optimum in subsequent genarations. Consider the case where the better offspring near the global optimum (See Figure 2.2-Individual A and Individual C). In such situation convergence of the better offspring to the global optimum is not altered by the proposed mutation profile. Hence the convergence characteristics not effected significantly. While the Diverse osspring undergo large mutation rate, these offspring have a opertunaty to paticipate in the group of better offspring in subsequent genarations.

Following figure shows the profile of convergance, when diverse offspring are introduced.

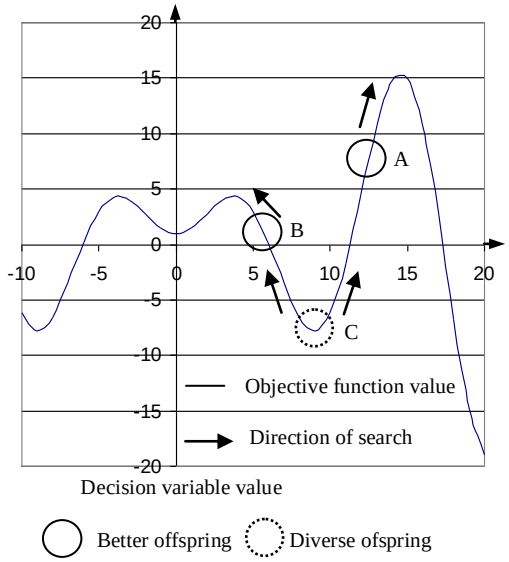


Figure 2.2: Illastraton of effect of diverse offspring to convergence characteristics when better offspring at alocal and the global optimum

Here the genetic algorithm was implemented using real value genes and all the genatic operations were implemented using real value mathamatical operations. The selection machanisam for crossover,mutation and insertion of diverse offspring was based on the trangular, and uniform probability density functions. Hence the algorithams of probability density functions also embadded in the program coding. Selection of individuals for the next generation is done by using a probability pool of individuals based on their fitness value and using select without replacement method to improve the diversity of the selected individuals by removing duplicates. For the crossover, intermidiate recombinaton method was used on real valued individuals to produce offspring near the parent individuals.

## 3. RESULTS

The results were obtained by first setting the initial parameters and benchmark objective functions in the genetic algorithm. For all the benchmark tests considered only the small population sizes. Here it is selected as 50. Generation gap is selected as 0.9 which is more commonly used in many benchmark tests for comparison purposes. Fraction of diverse offspring and probabilistic selection strategy was varied to find the effect of diverse offspring when

under go dynamic mutation. Fitness value calculated using the following equation which reaches zero at the optimum value of the objective function;

$$F(f_i(x_1, x_2,..)) = \ln\left(\left(f(x_1, x_2,...) - f_{final}(x_1, x_2,...)\right| + 1\right) \quad (3.1)$$

The results were averaged over ten runs and relevant standard deviation of results also calculated to find the consistency.

Following benchmark functions were used for benchmark tests of proposed genetic algorithm;

Table 3.1: Benchmark test functions for evaluation of performance of genetic algorithm.

| |
|---|
| Sine function<br>$F_{sine} = x.\sin(10\pi x) + 1.0$<br>This function has a global maximum in the interval $[-1,2]$ maximum at 2.85 with $x = 1.85$. |
| Step function<br>This function has two non linear parts and it has a highly bumpy surface. Global optimum is determined by the product term.<br>$F_{step} = 6 \times n + \sum_{i=1}^{n} \lvert x_i \rvert$<br>$n = 5, -5.12 \le x_i \le 5.12$ Global optimum=30 |
| Sphere function<br>This is a multimodal function contain many of local minima.<br>$F_{Sphere} = \sum_{i=1}^{n} x_i^2$<br>$n$ =30, $-5.12 \le x_i \le 5.12$<br>Global minimum of $F_{Sphere} = 0$ |
| Rastrigin function<br>This is a function with narrow ride sharp optimum and runs around a parabola.<br>$F_{Rastrigin} = n \times A + \sum_{i=0}^{n} \left(x_i^2 - A \times \cos(2\pi x_i)\right)$<br>$n$ =20, $-5.12 \le x_i \le 5.12$, A=10<br>Global minimum of $F_{Rastrigin} = 0$ |

Following abbreviations were used to label certain settings of the genetic algorithm.

FDUM*n.nn* : Conventional genetic algorithm with uniform mutation rate of *n.nn*.

FDTM*n.nn* : Proposed genetic algorithm with triangular mutation profile with maximum mutation rate of *n.nn* with diverse offspring.

Table 3.2: Average fitness and standard deviation of the results of ten runs after 1st, 5th, 15th, 19th, generations for deferent setting of genetic algorithms with sin function.

| Settings | Generation | Fitness | Std |
|---|---|---|---|
| FDUM0.01 | 1 | 0.248394 | 0.057690 |
| | 5 | 0.173386 | 0.007089 |
| | 15 | 0.021440 | 0.002481 |
| | 19 | 0.011101 | 0.000609 |
| FDUM0.10 | 1 | 0.258543 | 0.043948 |
| | 5 | 0.086964 | 0.009317 |
| | 15 | 0.030650 | 0.001715 |
| | 19 | 0.013149 | 0.000709 |
| FDTM0.50 | 1 | 0.265671 | 0.068048 |
| | 5 | 0.049012 | 0.006943 |
| | 15 | 0.013361 | 0.001757 |
| | 19 | 0.001766 | 0.000031 |
| FDTM1.00 | 1 | 0.243536 | 0.051435 |
| | 5 | 0.038944 | 0.004832 |
| | 15 | 0.009801 | 0.000962 |
| | 19 | 0.001350 | 0.000025 |

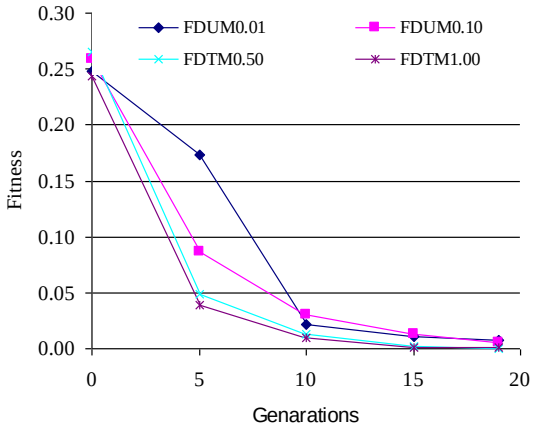


Figure 3.1: Graph of best fitness vs. number of generations for different mutation profile and rates of diverse offspring for sin function.

Table 3.3: Fitness and Standard deviation of the results after 1st, 10th, 20th, 35th generations for different settings of Genetic Algorithm with Step function

| Settings | Generation | Fitness | Std |
|---|---|---|---|
| FDTM1.00 | 1 | 2.678468 | 0.068056 |
| | 10 | 0.276812 | 0.007028 |
| | 20 | 0.023100 | 0.001820 |
| | 35 | 0.001621 | 0.000010 |
| FDUM0.10 | 1 | 2.581989 | 0.051294 |
| | 10 | 0.746785 | 0.011394 |
| | 20 | 0.305411 | 0.002253 |
| | 35 | 0.004849 | 0.000035 |
| FDTM0.01 | 1 | 2.628511 | 0.057704 |
| | 10 | 0.721327 | 0.007090 |
| | 20 | 0.397622 | 0.002523 |
| | 35 | 0.011183 | 0.000127 |

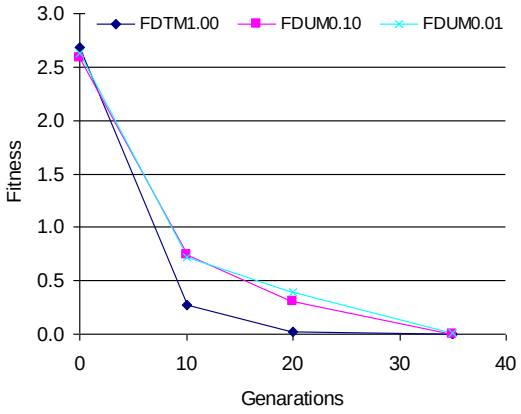


Figure 3.2: Graph of best fitness vs. number of generations for different mutation profile and rates of diverse offspring with Step function.

Table 3.4: Fitness and Standard deviation of the results after 1st, 50th, 100th, 150th, 200th generations for different settings of Genetic Algorithm with Sphere function

| Settings | Generation | Fitness | Std |
|---|---|---|---|
| FDTM1.00 | 0 | 5.554857 | 0.052358 |
| | 50 | 0.595676 | 0.020296 |
| | 100 | 0.018641 | 0.002374 |
| | 150 | 0.000023 | 0.000377 |
| | 200 | 0.000012 | 0.00013 |
| FDTM0.50 | 0 | 5.554857 | 0.072493 |
| | 50 | 1.188312 | 0.010258 |
| | 100 | 0.098824 | 0.002627 |
| | 150 | 0.026062 | 0.000127 |
| | 200 | 0.000427 | 0.000018 |
| FDUM0.10 | 0 | 5.554857 | 0.258851 |
| | 50 | 3.857881 | 0.070114 |
| | 100 | 3.85059 | 0.031505 |
| | 150 | 3.755615 | 0.013186 |
| | 200 | 3.510814 | 0.004343 |
| FDUM0.01 | 0 | 5.554857 | 0.067188 |
| | 50 | 2.244704 | 0.016211 |
| | 100 | 0.429929 | 0.002945 |
| | 150 | 0.089588 | 0.000647 |
| | 200 | 0.004895 | 0.000254 |

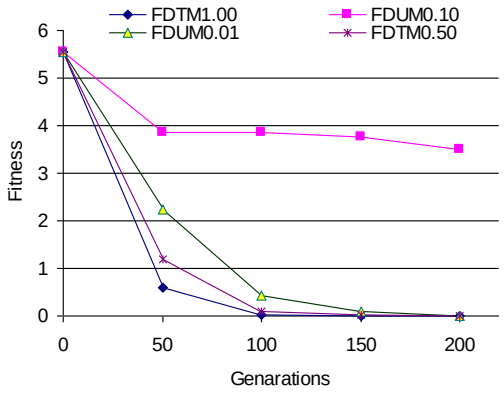


Figure 3.3: Graph of best fitness vs. number of generations for different mutation rates for diverse offspring with Sphere function.

Table 3.5: Fitness and Standard deviation of the results after 1st, 50th, 100th, 150th, 200th generations for different settings of Genetic Algorithm with Rasterign function

| Settings | Generation | Fitness | Std |
|---|---|---|---|
| FDTM1.00 | 1 | 5.554857 | 0.052358 |
| | 200 | 0.795676 | 0.020296 |
| | 400 | 0.003641 | 0.002374 |
| | 800 | 0.000023 | 0.000377 |
| | 1000 | 0.000012 | 0.000130 |
| FDUM0.01 | 1 | 5.884498 | 0.072181 |
| | 200 | 3.929961 | 0.055439 |
| | 400 | 1.126007 | 0.002386 |
| | 800 | 0.053126 | 0.000824 |
| | 1000 | 0.002302 | 0.000166 |

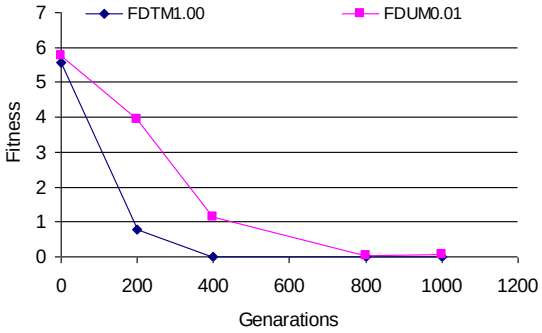


Figure 3.4: Graph of best fitness vs. number of generations for different mutation rates of diverse offspring with Rasterign function.

Graph in Figure 3.1 shows the significant improvement of convergence characteristics between the genetic algorithm with sin function using diverse offspring (FDTM1.00 and FDTM0.50) and conventional genetic algorithm (FDUM0.01 and FDUM0.10).It also shows that the high mutation rates can be used with diverse offspring without altering the convergence rate. Graph in Figure 3.2 shows similar behavior with the step function also.

It is clearly seen on the graph for sphere function in figure 3.3 FDUM0.10 that the even slightly higher mutation rates can not be used with the conventional genetic algorithm as it converges very slowly compared with genetic algorithms with diverse offspring(FDTM1.00 and FDTM0.50). Although the conventional genetic algorithm with lower mutation rate (FDUM0.01) in this figure shows better convergence rate, but the performance is less than the genetic algorithm with diverse offspring.

For the Rasterign function in figure 3.4, it shows significant improvement in convergence characteristics of genetic algorithm with diverse

offspring as compared with convergence characteristics of conventional genetic algorithm.

Additional to the above test simple random search algorithm also used with all above benchmark test functions. Result indicated that the convergence time is very unpredictable and totally random in nature. Also it has very large standard deviation in convergence time with high dimensional functions.

All above results were statistically analyzed for equivalence of initial population and the significance of final results after the number of generations specified in the test with different settings of the genetic algorithm. Statistical results shows that initial population of all above test were equal with 95% confidence level. Final result of the proposed algorithm with diverse offspring with higher mutation rate always shows significant improvement of the convergence characteristics with 95% confidence level.

## 4. CONCLUSION

Presented results show that the convergence characteristics and the probability of finding the global optimum is highly depended on the choice of the selection, ranking and other genetic operations . Dynamic mutation rate improve the convergence rate of the genetic algorithm. Introduction of diverse offspring improve the convergence characteristics by keeping the probability of finding the optimum in the search space non zero. Hence it is found from the results; introduction of diverse offspring with high mutation rates on diverse individuals does not alter the convergence characteristics compared to conventional genetic algorithms with higher mutation rates. Also introduction of diverse offspring increases the diversity of the population and significantly increase the searching performance of the genetic algorithm by preventing the risk of converging to a local optimum in considered search space. Result obtained for simple random search indicated that such algorithms are not suitable to use in applications where the running time is important and limited because of the unpredictability and large variability of convergence time. On the other hand, genetic operations implemented here were written using real value operations which gives predictable and strait forward implementation of program coding.

## 5. REFERENCES

[1] J. Rennard, “Introduction to Genetic Algorithms” in Journal of Information Technology Education, volume 6, 2007.

[2] M. K. Pakhira, “A Hybrid Genetic Algorithm using Probabilistic Selection”, Journal of the Institute of Engineers(India)84:23-30, May 2003.

[3] N. Zhu, Y.S.Ong, K.W. Wong and M.H. Lim, “Choice of Memes in Memetic Algorithm”, School of Computer Engineering, Nayangn Technological University, Singapore, July 2003.

[4] W. Rand and R. Riolo, “Shaky Ladders, Hyper-Defined Functions and Genetic Algorithms: Systematic Controlled Observations in Dynamic Environments”, University of Michigan, 2003.

[5] A. Chipperfield, P. Fieming, H. Pohlheim, C. Fonseka, “Genetic Algorithm Tool Box for Use with MATLAB”, Department of Automatic Control and Systems Engineering University of Sheffield.

[6] W. Bich et al, “Evaluation of Measurement Data–Supplement 1 to the Guideto the Expression of Uncertainty in Measurement–Propagation of distributions using a Monte Carlo method”, Final draft, Joint Committee for Guides in metrology JCGM, September 2006.

[7] J. Holland, A. Arbor, “Adaptation in Neural and Artificial Systems”, MI: University of Michigan, 1975.

[8] Microsoft Corp., MSDN: Visual C++ Help[Online].Available:http://msdn.microsoft.com/en-us/visualc/aa336440.aspx.

[9] D. Whitley, “A Genetic Algorithm Tutorial”, Computer Science Department, Colorado State University, September 2006.

[10] T. Weise, Global Optimization Algorithms – Theory and Application Version: 2009-06-26, pp20-231.

[11] R. Braune, S. Wagner, M. Arenzeller, “ Applying Genetic Algorithms to the Optimization of Production Planing in a Real-World Manufacturing Environment”, Institute of Systems theory and Simulation, John Kepler University, 2004.