Investigation into Mutation Operators for Microbial Genetic Algorithm

Samreen Umer
University College Cork, Western Road, Cork, Ireland

Keywords: Microbial Genetic Algorithm, Evolutionary Algorithms, Mutation.

Abstract: Microbial Genetic Algorithm (MGA) is a simple variant of genetic algorithm and is inspired by bacterial conjugation for evolution. In this paper we have discussed and analyzed variants of this less exploited algorithm on known benchmark testing functions to suggest a suitable choice of mutation operator. We also proposed a simple adaptive scheme to adjust the impact of mutation according to the diversity in population in a cost effective way. Our investigation suggests that a clever choice of mutation operator can enhance the performance of basic MGA significantly.

1 INTRODUCTION

The idea of using evolutionary principles for automated problems was first originated in the middle of 20th century. Dr. L.J Fogel, known as father of evolutionary programming, presented first evolutionary technique in his dissertation (Fogel et al., 1966). Later on, this area got significant attention and different dialects of this technology were evolved. Since then swarm based optimization and nature inspired algorithms marked their significance in computational sciences. In recent decades these techniques have gain much popularity in terms of optimization applications, system design and scheduling operations. Genetic algorithms are evolutionary computation techniques which mimics the evolution process to generate useful solutions for optimization problems (John, 1992). Many variants for genetic algorithms have been introduced to satisfy different constraints according to different situations. Here we will discuss about one of the variants of genetic algorithm called Microbial Genetic Algorithm (MGA).

Harvey introduced the idea of bacterial recombination or infection as a substitute to inheritance from parents in genetic algorithm (McCarthy, 2007) and called it as microbial genetic algorithm (Harvey, 2011). Motivation behind this variant was to produce a minimalist algorithm still containing all the characteristics of a true genetic algorithm. Since then, bacterial or microbial evolution based algorithms have been used successfully by many researchers and scientists. However any investigation on MGA's performance or its operators is not available in literature. We conducted this study to verify the behaviour of basic MGA and MGA with newer mutation operators to solve unconstrained optimization problems. We introduced a simple adaptive mutation scheme and results conclude that a clever selection of MGA operators can improve its performance notably.

2 MICROBIAL GENETIC ALGORITHM

MGA is a simpler variant of GA. Main difference between these two is different recombination operator. In MGA, initially parents are selected from generated population using a selection scheme and, are then evaluated on the basis of their fitness to mark a loser and winner. Unlike basic GA, MGA crossover genes in such a way that the winner is left intact and some genes are transferred to the loser with a defined crossover probability. In this way the elitism is maintained and assumingly good genes are transferred to the next generation. Mutation is then carried out on infected loser to keep the diversity in the population. The whole routine can be repeated until the desired results are achieved or up to maximum number of iterations allowed. Figure 2.1 illustrates one tournament or life cycle of MGA.

2.1 Generating Population

Like in any other evolutionary algorithm, the first step is to generate a potential population within domain

Umer, S..
Investigation into Mutation Operators for Microbial Genetic Algorithm.
In *Proceedings of the 7th International Joint Conference on Computational Intelligence (IJCCI 2015) - Volume 1: ECTA*, pages 299-305
ISBN: 978-989-758-157-1

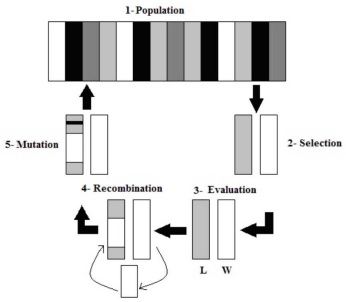

Figure 1: One MGA Cycle.

of the problem. Size and diversity of the population plays a crucial role in the performance of any algorithm. Larger the population ensures greater diversity but more computations.Small population can converge in less time but can also lead to premature convergence. Many studies have been conducted to generalize an optimal population size but mostly it depends on the dimension of the problem and number of maximum iterations allowed. Generally the population is generated from a uniform distribution within prescribed range and is unbiased.

2.2 Selecting Parents

A variety of selection technique is available for genetic algorithms. Some possibilities are stochastic universal sampling (Baker, 1985), Ranked selection (Whitley et al., 1989), Roulette wheel selection, and Truncation selection (Goldberg and Deb, 1991). However we have used ranked roulette selection as our selection scheme in experiments.

2.2.1 Ranked Roulette Selection

Roulette scheme of selection is one of the commonly used scheme among the researchers. In this method, the probability of selection P_i is assigned to the member i of population size n in accordance to their fitness value f_i such that

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{1}$$

For finding optimal minimums this formula can be manipulated as

$$P_i = \frac{\sum f - f_i}{\sum f} \tag{2}$$

But in case of non positive values unfortunately this equation can not work leaving us with the implication of some kind of scaling or scoring of fitness values. One possible solution to such problem was proposed by (Al Jadaan and Rao, 2008) as an improved selection operator. The population is first evaluated and sorted in accordance to the rank based on fitness value and then the probabilities are assigned using following formula

$$P_i = \frac{2R_i}{Pop(Pop+1)} \tag{3}$$

Where P_i is the probability of individual i with rank R_i from population with with population size Pop for being selected as a parent. A simple ranking scheme can be where rank of a member is its position in sorted population according to fitness.

The above scheme is used to select parents for each cycle or tournament. Parents are then evaluated and assigned as winner and loser for their survivability in next generation.

2.3 Recombination

In each cycle the population is replaced with new generation. The selection scheme mentioned above gives the parents to breed for children who will be introduced in population to make new generation. In conventional GA this has been the normal way of each life/death cycle where the parents are replaced with the children and thus genes are transferred vertically down to the next generation. However in Microbes, breeding is not in similar fashion. The organisms or microbes reproduce by binary fission and further exchange their genetic material via a process called bacterial conjugation. What happens in this phenomenon is that the fittest bacteria most commonly the one who has developed resistance to the antibiotics makes a contact with the weaker vulnerable bacteria and transfer its resistance in terms of plasmid to it. MGA uses this concept of parallel gene transfer to evolve instead of horizontal gene transfer in conventional GA. The fittest among the parents transfer its genes to the weaker thus the new chromosome is actually the weaker parent with some of its genes replaced by genes from the winner.

2.4 Mutation

Mutation is responsible for maintaining the diversity in population and preventing premature convergence. Extensive research and studies have been done already to provide different mutation operators and proved to be useful in most cases. Following operators are reviewed here for further use in experimentation ahead.

2.4.1 Uniform One Point Mutation

One random gene from the member which in this case is the infected loser, is replaced with a new gene. In case of binary MGA the gene is a bit so mutation means just flipping the bit. However in case of real number or integers, a new gene is generated from the possible range randomly. Uniform one point mutation is a technique in which at one random locus, the gene is replaced by a new one generated randomly from a uniform distribution.

2.4.2 Gaussian One Point Mutation

Instead of uniform distribution other distributions like Gaussian can be used for generating the new gene. A classical Gaussian mutation operator was developed by Rechenberg and Schwefel in which a scaled Gaussian normally distributed number is added to the previous value of gene (Back and Schwefel, 1993) . This method requires information about mean and variance of desired distribution to produce a random gene. Generally value of mean is set to be 0 with variance of 0.2 to 0.8. the formula for Gaussian density function with mean $\mu = 0$ and δ variance is given by

$$f(x) = \frac{1}{\sqrt{2\pi\delta^2}} e^{\frac{-x^2}{2\delta^2}} \tag{4}$$

2.4.3 Cauchy One Point Mutation

For some functions with relative minima that are far apart the small mutations of uniform or Gaussian distributions may lead to premature convergence to local minima. For this reason (Yao et al., 1999) devised an alternative to the Gaussian mutation by using a Cauchy distributed random number instead. Cauchy distribution with mean $\mu = 0$ is defined as

$$f(x) = \frac{t}{\pi(t^2 + x^2)} \tag{5}$$

Where t is the scaling factor and usually set as 1. This allows for larger mutation and can help prevent premature convergence and search the search space faster. However there is less probability of smaller mutations in the neighbourhood of the parent leading to a less accurate local search.

2.4.4 Forced Adaptive Guassian Mutation

Experiments suggest that MGA converges faster in earlier stages but with time as population tends to be more similar, the same mutation operator does not remain as effective. Many researchers have proposed adaptive or forced mutation operators for enhancing its over all performance. In (Hatwagner and Horvath, 2012) a new forced mutation operator has been introduced for bacterial evolutionary algorithm based on relationship between diversity of population and variance of mutation operator. However calculating diversity in each iteration incurs additional computational overhead and delays. Inspired by their technique, we designed a mutation operator where we set thresholds to calculate diversity in the population and adjust mutation accordingly. The diversity of population is calculated using formula given in (Miller and Goldberg, 1995).

$$d_{i,j} = \sqrt{\sum_{k=1}^{g} \left(\frac{x_{i,k} - x_{j,k}}{X_{k,max} - X_{k,min}}\right)} \tag{6}$$

$$D = \frac{1}{Pop - 1} \sum_{i=1}^{Pop} d_{i,best} \tag{7}$$

Any mutation scheme can be coupled with this technique however we used gaussian scheme where the variance is controlling parameter of mutation. Objective for designing this operator is to improve the diversity D in earlier evaluations and tune to fine position in the later stages. So when the population is generated it usually has higher diversity at that point we can directly relate the variance σ with D allowing MGA routine to get variety in its early generations. With higher variance we observed no improvement in this situation where as low variance in such condition showed better refine. Using following formula where Variance σ increases with diversity up to a threshold value lets say $D = 0.5$ and then start decreasing linearly .

$$\sigma = \alpha.min(D, 1 - D) \tag{8}$$

α is the scaling constant with values $0 \leq alpha \leq 1$. σ obtained is then used in equation (4) to facilitate gaussian mutation.

2.5 Replacing Previous Generation with New One

Now comes the turn to introduce the infected and mutated member into the population to proceed for next cycle. In basic original GA, usually either both of the selected parents are replaced with new crossed and mutated offsprings or the worst of whole population is replaced with the new offsprings. However to maintain the elitism by keeping the winner intact and also avoid losing all of the genes from loser, in basic MGA loser chromosome is replaced with the new infected and mutated version of loser. This is easy to implement and effective as well. This keeps in consistency with the underlying intuition of microbial evolution where evolution is brought without death of parents.

3 EVALUATION

Performance of any optimization algorithm is measured through some characteristic features like its robustness and precision. A set of standard benchmark problems are usually used to analyze these features for any algorithm. These functions are well known in literature, and their qualitative properties and global extremes are also known. The notion of robustness may refer to a set of characteristics itself. This tells how good the algorithm can perform with increasing number of dimensions and difficulty as well as how timely it can converge to optimal solution. The term convergence may be interpreted in two meanings, either all the population becoming just uniform at any suboptimal location or when the evaluation hits the desired optimal results. Both these situations are closely related but different from each other. Precision of the algorithm tells how close the results are to desired optimal solution. A fast, robust and, precise algorithm is an objective of optimization researcher. A number of simulations are carried out to testify the proposed suggestions. These simulations are run on MATLAB 7.1 to give numerical results and graphical representation of algorithm performances. We have used a set of standard benchmark from literature (Ortiz-Boyer et al., 2005) for our simulations categorized under three subsets as follows.

- Unimodal multidimensional convex problems
- Multimodal two dimensional problems
- Multimodal multidimensional problems

Unimodal multidimensional convex problems include some cases causing poor or slow convergence to a single global extreme. Multimodal problems may have large number of local extremes making a multidimensional problem harder to locate the global optima. Increasing number of dimensions further increases the hardness of the problem but most of the real practical problems desired to be modelled usually are multidimensional. For evaluation, we run monte carlo simulations to find suitable controlling parameters like probabilities and constants and then average the results from 100 simulations for each problem and variant of MGA.

3.1 Simulations Results

In this section, we have presented simulation plots for MGA with different mutation operators. We have used ranked roulette selection scheme in all simulations and different mutation operators for each optimization problem. The results plotted are mean values of 100 simulations for each evaluation. Probability of mutation and infection have been chosen after extensive experimental investigation. Table 1 provides chosen values for probabilities of infection *Pi* and probability of mutation *Pm* for each problem and also states the best solution achieved over 10000 iterations and $n = 2, 30$. MGA with adaptive mutation is also compared with basic GA over 5000 iterations using MATLAB GA toolbox with suggested values of parameters from literature and ranked roulette selection scheme as in MGA. Comparison with GA is given in table 3 in appendix and shows that MGA with adaptive mutation is highly comparable with basic GA and even outperforms GA in many problems in terms of finding better minima specially in problems with less variables or dimensions.

Table 1: Chosen Parameters and Best Achieved Solutions in 5000 iterations.

Problem	Best Minima	Pi	Pm
DeJong First	0.0002	0.7	0.7
Axis Parallel Ellipsoid	0.0025	0.7	0.7
Rotated Hyper Ellipsoid	0.0020	0.7	0.7
Sum of Different Powers	0.0092 x10^{-4}	0.7	0.7
Rosenbrok Valley	0.1081	0.7	0.7
Rastrigins	0.0329	0.7	0.7
Schwefel	-1256.9	0.6	0.8
Griewangk	0.0566x10^{-6}	0.7	0.8
Ackley	0.0104	0.6	0.8
Branin	0.3978	0.5	0.75
Goldstein Price	3	0.6	0.4
Six Hump Camel Back	-1.036	0.6	0.75
DeJong Fifth	0.998	0.6	0.75

3.1.1 Unimodal Multidimensional Problems

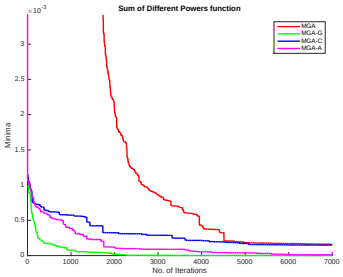

Figure 2: Sum of different powers function.

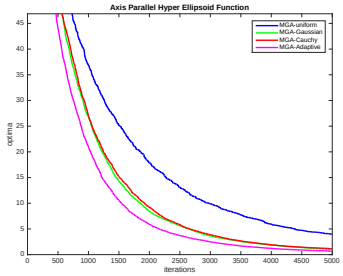

Figure 3: Axis parallel hyper ellipsoid function.

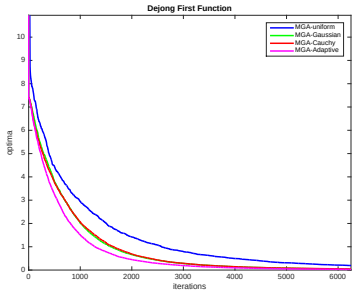

Figure 4: Dejong first function.

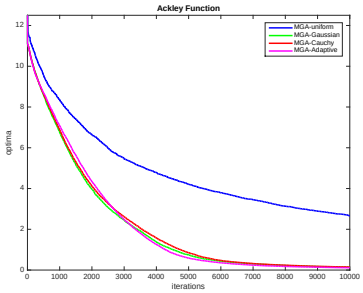

Figure 8: Ackley function.

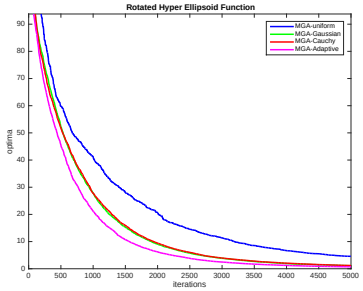

Figure 5: Rotated hyper ellipsoid function.

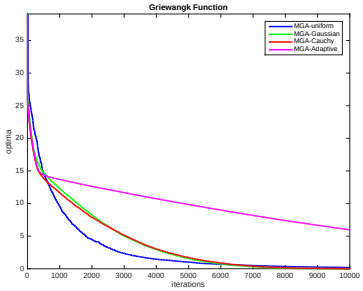

Figure 9: Griewangk function.

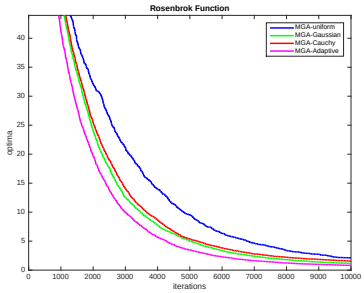

Figure 6: Rosenbrok valley function.

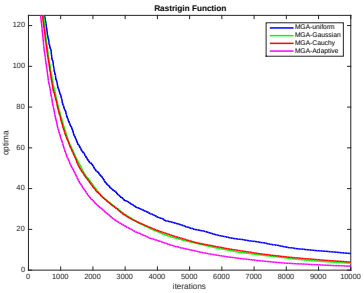

Figure 10: Rastrigin function.

3.1.2 Multimodal Multidimensional Problems

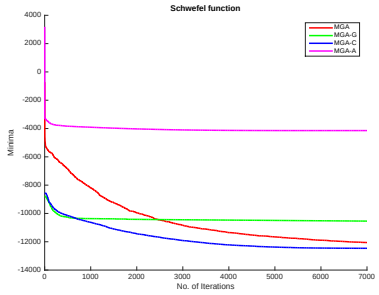

Figure 7: Schwefel function.

3.1.3 Two Dimensional Multimodal Difficult Problems

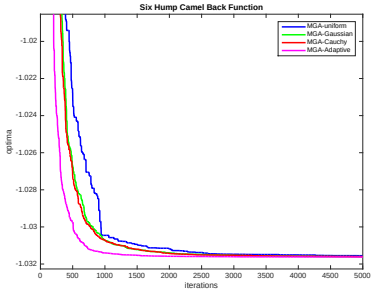

Figure 11: Six hump camel back function.

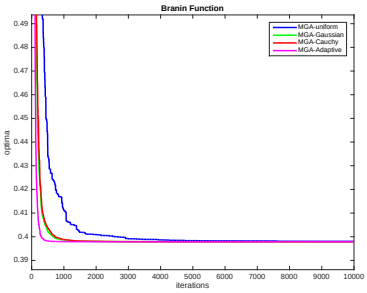

Figure 12: Branin function.

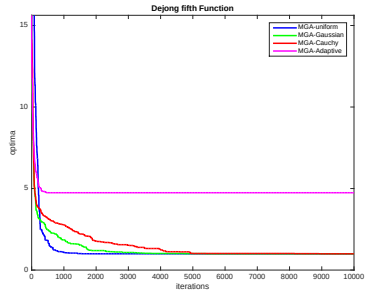

Figure 13: Dejong fifth function.

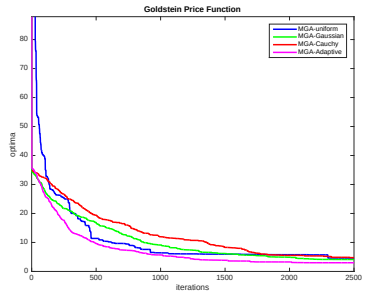

Figure 14: Goldstein price function.

4 CONCLUSIONS

In this paper, we have analyzed performance of MGA in minimizing unconstrained complex problems using commonly used mutation operators. We have also suggested a simple adjustment to adapt the mutation according to the diversity in population to be more effective. Comparison of different variants of MGA might be helpful for readers in choosing the suitable variant according to their objective problem. MGA with our adaptive mutation has shown promising behaviour in general and comparison with basic GA verifies its potential as a useful optimization technique.

REFERENCES

Al Jadaan, Omar, L. R. and Rao, C. (2008). Improved selection operator for genetic algorithm. *Theoretical and Applied Information Technology*, (4.4).

Back, T. and Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). Artificial intelligence through simulated evolution.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, 1:69–93.

Harvey, I. (2011). The microbial genetic algorithm. In *Advances in artificial life. Darwin Meets von Neumann*, pages 126–133. Springer.

Hatwagner, F. and Horvath, A. (2012). Maintaining genetic diversity in bacterial evolutionary algorithm. *Annales Univ. Sci. Budapest, Sec. Comp*, 37:175–194.

John, H. (1992). Holland, adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence.

McCarthy, J. (2007). What is artificial intelligence. *URL: http://www-formal. stanford. edu/jmc/whatisai. html*, page 38.

Miller, B. L. and Goldberg, D. E. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212.

Ortiz-Boyer, D., Hervás-Martínez, C., and García-Pedrajas, N. (2005). Cixl2: A crossover operator for evolutionary algorithms based on population features. *J. Artif. Intell. Res.(JAIR)*, 24:1–48.

Whitley, L. D. et al. (1989). The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *ICGA*, pages 116–123.

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

APPENDIX

Table 2: Test Functions Used.

Name	**Min**	**domain**
Unimodal Multidimensional Problems		
De Jong First	0	$-5.12 \leq x_i \leq 5.12$
Axis Parallel Ellipsoid	0	$-5.12 \leq x_i \leq 5.12$
Rotated Hyper Ellipsoid	0	$-65.54 \leq x_i \leq 65.54$
Sum of Different Powers	0	$-1 \leq x_i \leq 1$
Rosenbrocks Valley	0	$-2.048 \leq x_i \leq 2.048$
Multimodal Multidimensional Problems		
Rastrigins	0	$-5.12 \leq x_i \leq 5.12$
Schwefel	-418.98n	$-500 \leq x_i \leq 500$
Grienwangk	0	$-600 \leq x_i \leq 600$
Ackley	0	$-32.76 \leq x_i \leq 32.76$
Difficult Two dimensional Multimodal Problems		
Branin	0.397887	$-5 \leq x_i \leq 15$
Goldstein Price	3	$-2 \leq x_i \leq 2$
Six Hump Camel Back	-1.036	$-3 \leq x_i \leq 3$
De Jong Fifth	0.998	$-65.54 \leq x_i \leq 65.54$

Table 3: Comparison of GA with MGA-Adaptive over 5000 iterations.

Problem	**MGA-Adaptive**			**GA**		
	Minima Achieved	P_i	P_m	Minima	P_c	P_m
DeJong First	0.0002	0.7	0.7	**0.0001**	0.8	0.2
Axis Parallel Hyper Ellipsoid	**0.0025**	0.7	0.7	0.0028	0.8	0.2
Rotated Hyper Ellipsoid	**0.0020**	0.7	0.7	0.0021	0.8	0.2
Sum of Different Powers	**0.0092 x10**$^{\{-4\}}$	0.7	0.7	0.0009	0.8	0.2
Rosenbrok Valley	0.1081	0.7	0.7	**0.0541**	0.8	0.2
Rastrigins	**0.0329**	0.7	0.7	0.004	0.8	0.1
Schwefel	-1256.9	0.6	0.8	**-1494.0**	0.8	0.1
Griewangk	0.0566x10$^{\{-6\}}$	0.7	0.8	**1.364x10**$^{\{-7\}}$	0.8	0.1
Ackley	0.0104	0.6	0.8	**0.0014**	0.8	0.1
Branin	**0.3960**	0.5	0.75	0.3979	0.6	0.2
Goldstein Price	**3.0000**	0.6	0.4	3.002	0.6	0.2
Six Hump Camel Back	**-1.0360**	0.6	0.75	-1.032	0.6	0.2
DeJong Fifth	**0.9980**	0.6	0.75	0.998	0.6	0.2