OPEN ACCESS

International
Journal
Of Modern Engineering Research (IJMER)

A Genetic Algorithm on Optimization Test Functions

Emmanuel S. Adabor[1*], Joseph Ackora-Prah[2]
[1]*School of Technology, Ghana Institute of Management and Public Administration, Accra, Ghana*
[2]*Department of Mathematics, Kwame Nkrumah University of Science and Technology, PMB, Kumasi, Ghana*
**Corresponding Author: emmanuelsadabor@gmail.com*

ABSTRACT: *Genetic Algorithms (GAs) have become increasingly useful over the years for solving combinatorial problems. Though they are generally accepted to be good performers among meta-heuristic algorithms, most works have concentrated on the application of the GAs rather than the theoretical justifications. In this paper, we examine and justify the suitability of Genetic Algorithms in solving complex, multi-variable and multi-modal optimization problems. To achieve this, a simple Genetic Algorithm was used to solve four standard complicated optimization test functions, namely Rosenbrock, Schwefel, Rastrigin and Shubert functions. These functions are benchmarks to test the quality of an optimization procedure towards a global optimum. We show that the method has a quicker convergence to the global optima and that the optimal values for the Rosenbrock, Rastrigin, Schwefel and Shubert functions are zero (0), zero (0), -418.9829 and -14.5080 respectively.*

Keywords: *Genetic Algorithms, Combinatorial problems, multimodal optimization problems, meta-heuristic algorithms*

I. INTRODUCTION

Global optimization techniques have been applied to real life problems such as determining the best solution from a set of possible solutions. Algorithms for such purpose are deterministic if there is a clear relation between the characteristics of the possible solution and the fitness for a given problem. On the other hand, probabilistic algorithms solve problems where the relation between a solution and fitness is complicated and the dimensionality of the search space is very high. Such algorithms employ heuristic approaches: Monte Carlo class of search algorithms. Several approaches that can determine the maximum or minimum of optimization problems exist. An extremely basic approach is the random search approach that randomly explores the search space by randomly selecting any solution and testing with other solutions [1,2,3]. Although this allows a good number of solutions to be tested, it does not guarantee global optima. However, it is successfully applied when employed in other algorithms [1].

The gradient based methods can be used to solve optimization problems in which the objective functions are smooth [1]. These employ local search by finding a gradient vector or Hessian matrix for generating better minima of solutions to a problem. Although these methods are tractable and search in shorter times, they are only applied to functions which are smooth.

Furthermore, there are greedy algorithms for problems where objective functions are not continuous [4]. These methods explore only neighbouring solutions to the current solution for optimal solution. This property makes them unreliable since they could be trapped on local optima. However, repeating the greedy algorithms with different starting solutions have been found to produce better results [1].

In the Simulated Annealing search method, a solution is accepted to be better than the current solution if it improves the objective function [5,6,7]. If a solution does not improve the objective function, then it is accepted with a probability. Even though the Simulated Annealing method produces very good results, Genetic Algorithms (GAs) have been found to be better because they test populations of solutions instead of a single solution or point in the Simulated Annealing method [8,9].

Recently, GAs have become dynamic and easy to use due to further research. For instance, Naoki et al., (2001) utilized the thermo-dynamical genetic algorithms (TDGA), a genetic algorithm which maintains the diversity of the population by evaluating its entropy, for the problem of adaptation to changing environments [10]. A modification of genetic algorithm is used in the problem of wavelength selection in the case of a multivariate calibration performed by Partial Least Squares (PLS). The Genetic Algorithm sets weights to balance the components out to solve the problem by an automated approach [11].

Wiendahl and Garlichs presented a graphical-oriented decision support system for the decentralized production scheduling of assembly systems using a Genetic Algorithm as the scheduling algorithm [12]. In the

past, GAs have been applied to solve resource constrained project scheduling problem (RCPSP) to minimize make-span subject to precedence constraints and resources availability [13]. In these approaches, resources are considered renewable as well as single mode of conducting every activity. A new permutation of encoding scheme was designed in the GA to inherit all the advantages of both the new permutation-based encoding scheme and the priority-based encoding schemes which were introduced [13].

In Biological Sciences, GAs have been applied to medical optimal control problem in immunology [14]. The author of such application provided the main strands of GA theory. Dynamic programming methods have been applied to solve Multiple Sequence Alignment in Molecular Biology leading to exponential time complexity. This has necessitated the use of GAs to solve the Multiple Sequence Alignment problem resulting in improved results over the dynamic programming methods [15]. The determination of the risk factors of the Alzheimer's disease provides insights into the disease. Although statistical approaches have been used for this purpose, GAs have been more useful and efficient for identifying these factors since they are able to search combinations of variables when search space is large [16].

Recent developments in design of experiments in Statistics have found GAs to be effective tools for optimum designs. In particular, desirable experimental designs that exploit the intrinsic capabilities of GAs have been produced. See Lin et al. (2015) for instances [17]. GAs have also been applied to optimizing pipe networks [18], Resource Economics problem [19], hydrological model (GASAKe) [20], surface electromyography signal [21], IIR digital filters [22] and Bayesian Networks [23].

The justifications for applying GAs are not well understood even with the numerous applications. Therefore, in this paper, we show the suitability of GAs for solving complex problems involving more than one variable with multiple modes and their ability to escape from being trapped in local optimal solutions. These are shown in the examination of the performance of GAs on special optimization test functions. The test functions are Rosenbrock function, Schwefel function, Rastrigin function and Shubert function.

II. METHOD

2.1 Simple Genetic Algorithm (GA)

The GA handles a population of possible solutions represented by a set of chromosomes. It proceeds by coding all the possible solutions into chromosomes before determining reproduction operators. They are applied directly on the chromosomes, and are used to perform mutations and recombination. Each chromosome has an associated value that corresponds to the fitness of the solution [1].

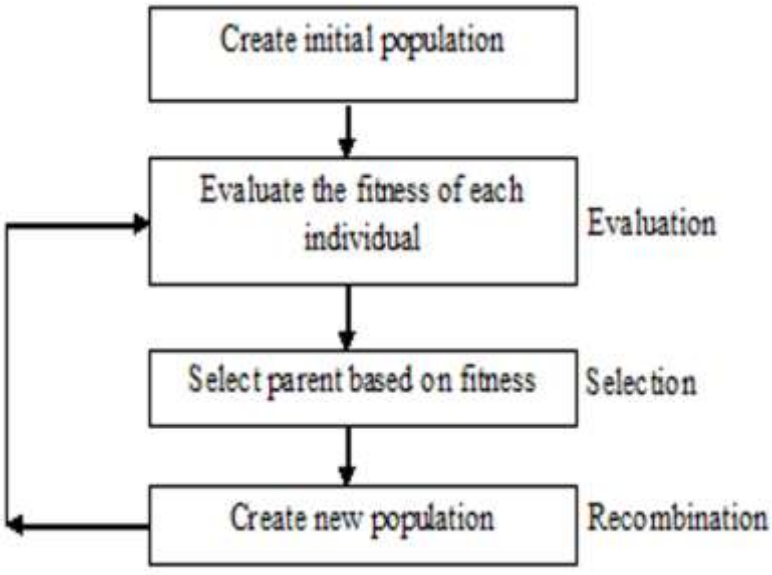

Figure 1. Overview of Genetic Algorithm

2.1.1 Encoding and Selection

Encoding is representing individual genes using bits, numbers, trees, arrays, lists or any other objects. The binary string uses bit string made up of zero (0) and one (1). Fig. 2 is an example of a binary string encoding. Every bit string is a solution but not necessarily the best solution.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure 2. A Population of four chromosomes

Selection involves choosing individuals in the population for the next generation. It also determines number of offspring that will be created by each. The purpose of the selection is to derive fitter individuals in order to improve solution. The level of the fitness function determines the chances of an individual to be selected.

In order to select fitter chromosomes for the mating pool, the proportional fitness (roulette wheel) selection method which selects parent by probabilities were applied. This means that the probability of selecting an individual is directly proportional to its fitness. This is a linear search through a *roulette wheel* with the slots in the wheel weighted in proportion to the individual's fitness values. The population is explored until a target value is reached. The *roulette wheel* constructs the relative fitness and cumulative fitness of chromosomes in its first stage. The relative fitness of each chromosome for a maximization problem is

$$w_i = \frac{f_i}{\sum_{k=1}^{n} f_k} \qquad (1)$$

where f_k is the fitness of the *k-th* chromosome, and *n* is population size while the expression for a minimization problem is:

$$w_i = \frac{F_i}{\sum_{k-1}^{n} F_k} \qquad (2)$$

and

$$F_i = \left(f_{max} - f_i\right) + 1 \qquad (3)$$

where f_{max} is the maximum fitness of all chromosomes and F_k is the reverse magnitude fitness.

The cumulative fitness (c_j) of the *j-th* chromosome is given by Equation (4):

$$c_j = \sum_{i=1}^{j} w_i \qquad (4)$$

In the second stage, a random number, r_j, is chosen and if $r_j > c_j$ then the *j-th* chromosome is selected.

2.1.2. Crossover

Crossover is taking two parent solutions and producing children solution from them. Better offspring are created by applying crossover to the mating pool to enable the GA to converge. This involves randomly choosing some crossover points, copying every element before this point from the first parent and copying every element after the crossover point from the other parent. In particular, we choose two crossover points and the contents between these points are exchanged between two mated parents. The two points crossover enhance the performance of the GA by maintaining the building blocks while exploring the search space. The power of crossover operation to thoroughly explore the search space diminishes as the GA approaches convergence.

A crossover probability of 1 implies all offspring must be made by crossover. If it is 0, then a whole new generation is to be generated from exact copies of chromosomes from old population. The new

chromosomes are intended to bear the good parts of old chromosomes so that the new chromosomes are better/fitter individuals. This informed the choice of 80% crossover probability in this research.

2.1.3. Mutation

Mutation maintains genetic diversity in the population generated from crossover. This enables the algorithm to escape from being trapped on local optimal solutions. The binary representation of the solutions suffices that a simple mutation of each gene with a small probability would be needed. A Bit flipping mutation was adopted in implementing the algorithm. This involves changing 0 to 1 and 1 to 0 based on a mutation chromosome generated. As illustrated in Fig. 3, a parent is considered and a mutation chromosome is randomly generated. For a 1 in mutation chromosome, the corresponding bit in parent chromosome is flipped (0 to 1 and 1 to 0) and child chromosome is produced.

Parent 1	**1 0 1 1 0 1 0 1**
Mutation chromosome	**1 0 0 0 1 0 0 1**
Child	**0 0 1 1 1 1 0 0**

Figure 3. Mutation flipping

The mutations to existing parents or chromosomes are performed with a mutation probability. If mutation probability is 100%, the entire chromosome is changed, if it is 0%, nothing is changed. Thus, offspring are generated immediately after crossover in case the mutation probability is zero. However, if mutation is performed, one or more parts of a chromosome are changed. The mutation probability used to examine the simple GA in this work is 0.2 permitting diversity in the population of solutions.

2.1.4. Replacement

Once offspring are produced, the parent population is assessed with the offspring for possible replacement to form new set of solutions. The weaker parents are replaced in a weak replacement procedure to ensure optimum performance of the GA. This involves replacing a weaker parent by a strong child. For instance given two parents and two offspring, only the fittest two of the four individuals, parent or child form part of the new population. Replacing the weak parents with fitter children improves the fitness of the candidate population solution.

2.1.5. Termination of Search

The algorithm converges after a specified number of generations, a fixed time and a zero change in the fitness value over a specified number of consecutive generations.

2.2 Test Functions

Test functions enable the evaluation of optimization algorithms with respect to their convergence rate, robustness and precision. These functions belong to the class of multi-modal and multidimensional functions with large number of extrema. Examples of such functions are the Rosenbrock function [24], the Rastrigin function [25,26], the Schwefel function and the Shubert function [27].

2.2.1. Rosenbrock Function

The Rosenbrock function (Fig. 4) given by Equation (5), has a global optimum that lies inside a long, narrow, parabolic-shaped flat valley.

$$f(x,y) = (1-x)^2 + 100\left(y - x^2\right)^2, (x,y) \in [-2.048, 2.048] \quad (5)$$

The convergence to the global optimum by search algorithms is difficult and hence this problem has been frequently used to test the performance of optimization algorithms. The minimum point causes a lot of problems for search algorithms such as the method of steepest descents which will quickly find the entrance to the valley and then spend several times searching from one side to the other. Such algorithms converge very slowly toward the minimum.

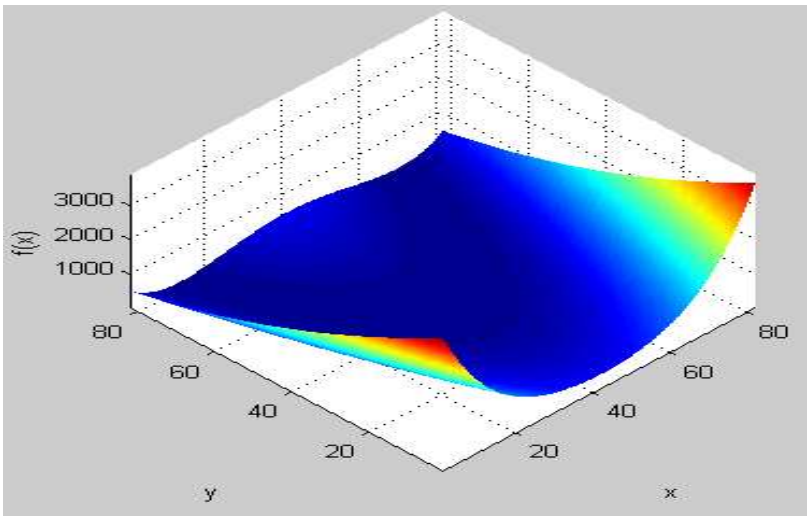

Figure 4. Overview of Rosenbrock function

2.2.2. Rastrigin Function

This function is highly multi-modal as illustrated in Fig. 5. Nevertheless, the locations of local minima are regularly distributed across the plot of the function.

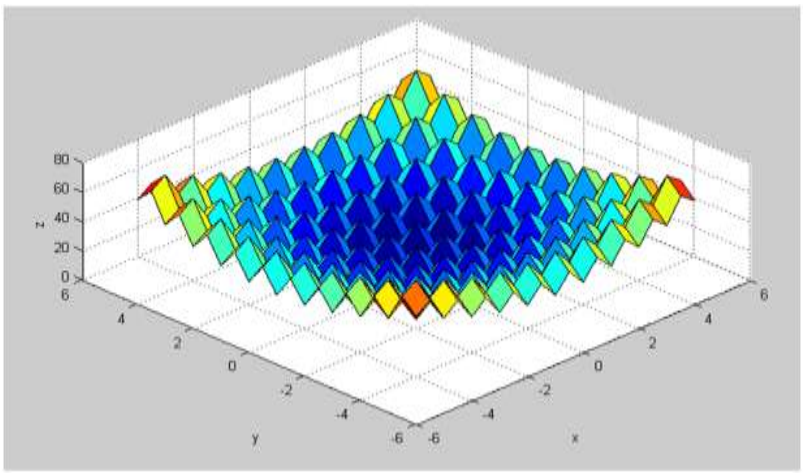

Figure 5. Overview of Rastrigin function

The Rastrigin function is given by Equation (6).

$$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right], x_i \in [-5.12, 5.12]. \quad (6)$$

2.2.3. Schwefel Function

The Schwefel function is given by Equation (7). It is highly deceptive in that the global minimum is geometrically distant over the parameter space. That is the consecutive best local minima are distant from one another (See Fig. 6). This makes search algorithms falter in converging to the direction of the global minimum of the function.

$$f(x) = \sum_{i=1}^{n}\left[-x_i \sin\left(\sqrt{|x_i|}\right)\right], x_i \in [-500, 500] \quad (7)$$

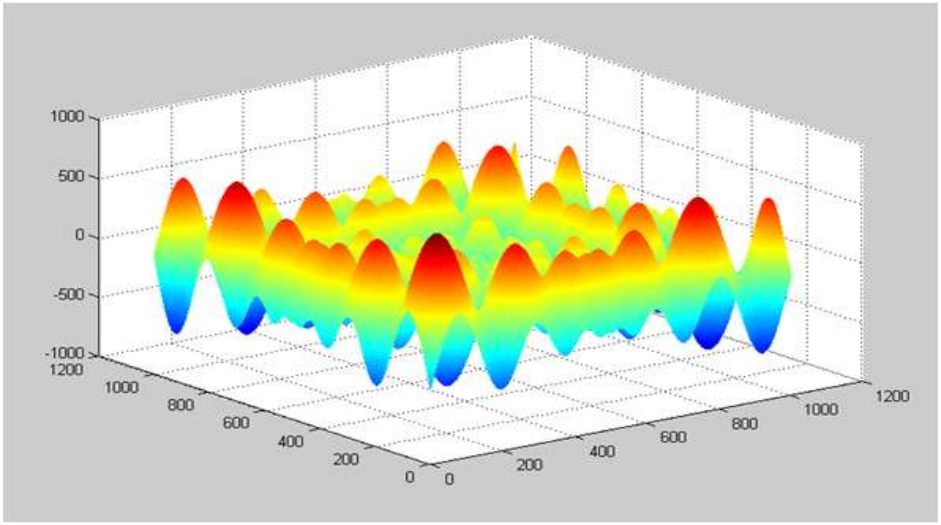

Figure 6. Overview of Schwefel function

2.2.4. Shubert Function

The Shubert function is a multi-variable function with multiple modes making it difficult for several algorithms to easily obtain the optimal solution. The complex nature of the function is illustrated in Fig. 7. The Shubert function can be written as a function of two variables shown in Equation (8).

$$f(x,y) = -\sum_{i=1}^{5} i\cos[(i+1)x+1]\sum_{i=1}^{5} i\cos[(i+1)y+1], \; (x,y) \in [-5.12, 5.12]. \tag{8}$$

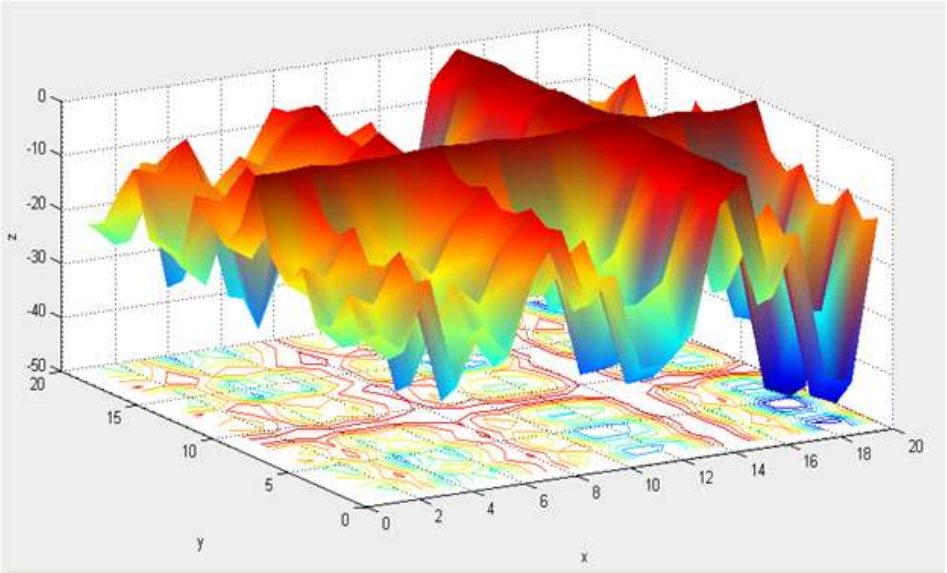

Figure 7. Overview of Shubert function

III. RESULTS

3.1. Performance Indicators of Algorithms

The performance of the simple GA is measured by the values of the optimal solutions. The manual implementation of the GA is infeasible due to the huge population size that is generated at each stage of the algorithm. Therefore, the minima of the test functions were solved with MATLAB codes (MathWorks Inc., 2006) and the results were examined to ascertain the performance of the algorithm. For optimum performance of the algorithm, the initial population size was limited to 50. This ensures an effective exploration of search space and quick convergence.

Furthermore, a maximum of 100 generations was specified since the generations hereafter do not produce

significant difference in the optimal values for each function. Nevertheless, the algorithm stops when there are no improvements in the optimal value after 50 consecutive generations. This accounts for the limit on the stall generations. It must, however, be stated that there is no perfect way for setting parameters except by experimental training which have been conducted [28]. Therefore, the choice of these parameters and the others are well justified.

3.2. Performance on Test Functions

The minima of the Rosenbrock function, Rastrigin function, Schwefel function and Shubert function are presented in Table 1. The minima of all the functions were reached at the end of 51 generations and at a point where the average change in fitness values were less than the tolerance, 1×10^{-6} .

Table 1. Results of Mimina of functions

Function	Number of decision Variables	Optimal solution	Optimal value
Rosenbrock	2	1.0070, 1.0140	0.0000
Rastrigin	1	0.0000	0.0000
Rastrigin	5	0.0125, 0, -0.0*, 0.001, -0.0*	0.0309
Schwefel	1	420.9618	-418.9829
Schwefel	10	$10^{114}\times V$	-4.7620×10^{114}
Shubert	1	-0.2001	-14.5080
Shubert	3	-0.1887, -0.2017, -0.2258	-3.0246×10^{3}

Vector, V = (-0.0*,0.0*,-0.0*,-0.0*,-0.0*,0.0*,-0.0*,-4.9300,-0.0*,0)
* value is not zero

The simulations of the search for best fitness values were investigated. In search for best fitness values from the initial values, it was revealed that though the fitness values for the population differed greatly, the GA was able to identify the best value at each stage of the search. For instance, the search for the minimum of the Rosenbrock function identified a mean fitness value of 30 though the optimal value was about zero (0) in the first generation. Details of similar observations at each stage of the search are presented in Fig. 8.

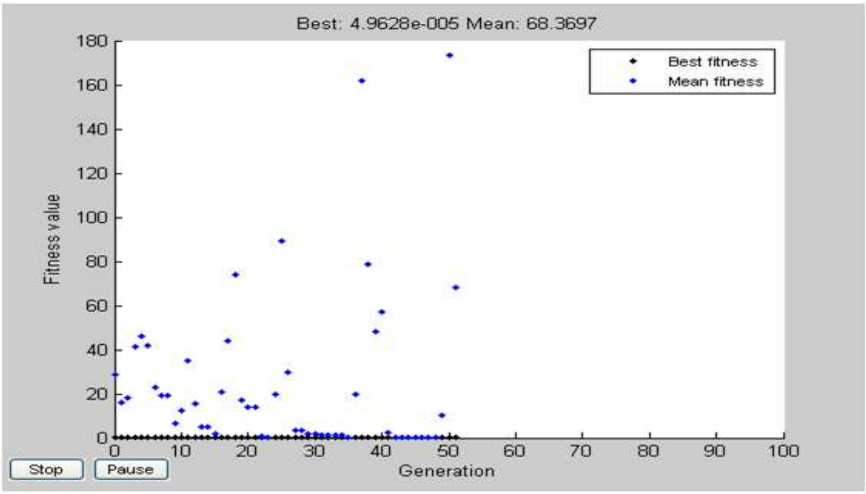

Figure 8. A simple Genetic Algorithm search for optimal value of the Rosenbrock function. The best fitness value overall the generations was approximately zero (0) and the mean fitness value over all generations was 68.3697.

Although the simple Genetic Algorithm began with a poor initial randomly generated population of solutions having achieved a mean fitness value of 9.6 for the Rastrigin function, it achieved the expected optimal value from the beginning of the search over 51 generations. Details of these observations and the thorough search through the regularly distributed modes of the Rastrigin function are presented in Fig. 9.

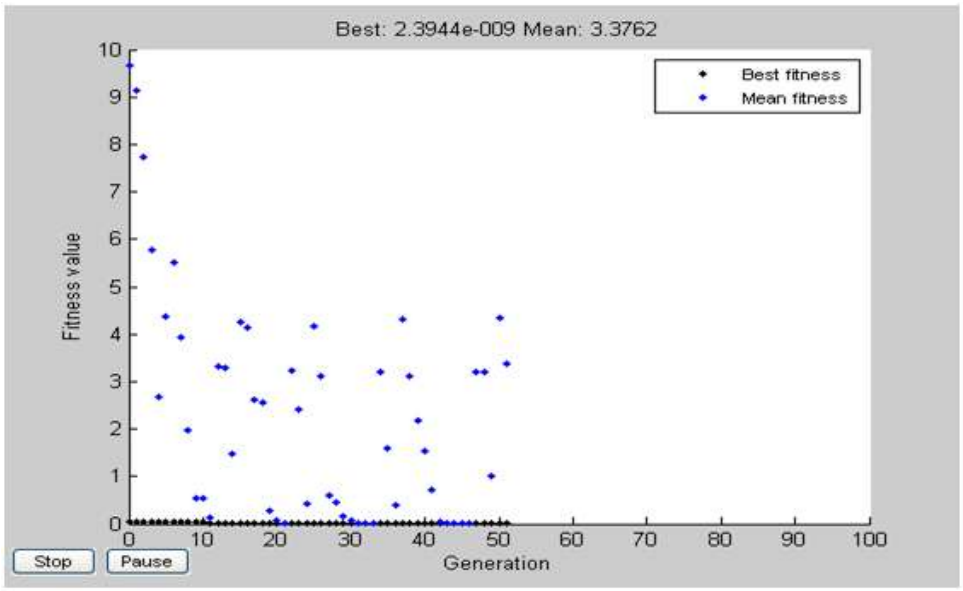

Figure 9. The blue plot is mean fitness for generation and the black plot is the best fitness for generation. A simple Genetic Algorithm search for optimal value of the Rastrigin function.

The nature of the simple GA search for the optimal value of the Schwefel function was also investigated. It was revealed that though an optimal value of -418.9829 was achieved after 51 generations, the best initial value at the first stage of the search was about zero which coincided with the mean fitness value. The progress of the search for the minimum of the Schwefel function from the initial optimal value to the final optimal value at the end of 51 generations are presented in Fig. 10.

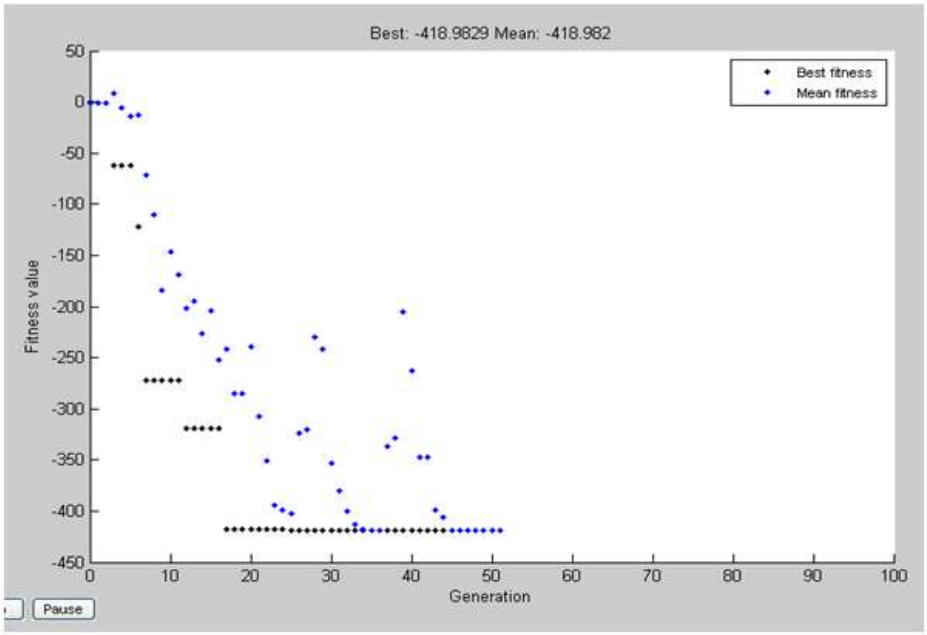

Figure 10. A simple Genetic Algorithm search for optimal value of the Schwefel function. The blue plot is mean fitness for generation and the black plot is the best fitness for generation.

Besides these exploratory abilities shown by the GA, it was able to identify the optimal value of the Shubert function. In particular, after 38 generations, the mean fitness value of population coincides with the optimal value at subsequent generations. Fig. 11 presents details of the search for the best minimum of the Shubert function.

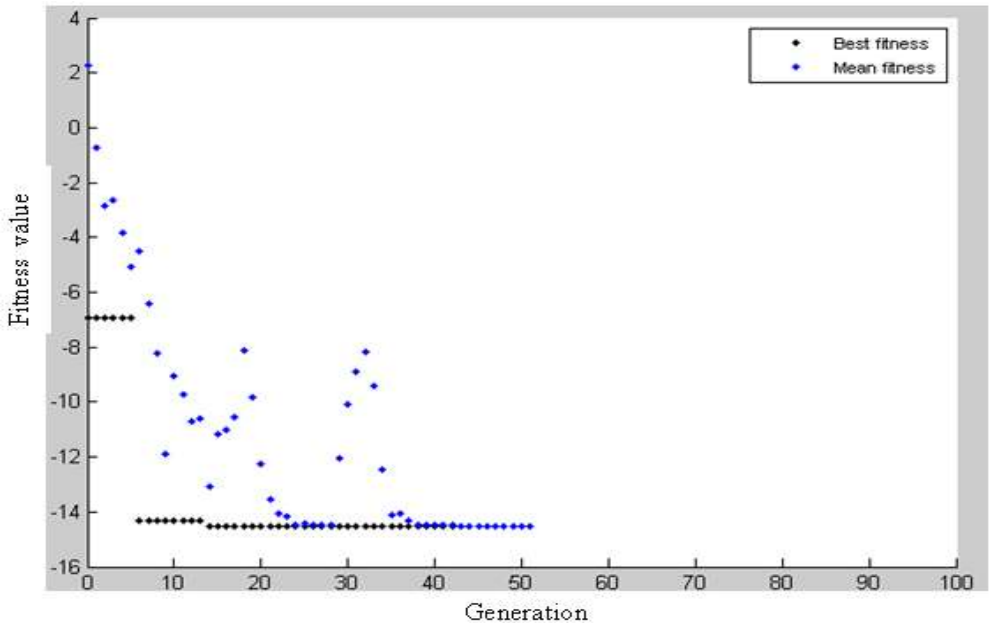

Figure 11. A simple Genetic Algorithm search for optimal value of the Shubert function. The blue plot is mean fitness for generation and the black plot is the best fitness for generation.

IV. DISCUSSION

The Rosenbrock function had the optimal value of zero which occurred at the point (1.0070, 1.0140). The results of the Rosenbrock function confirms results in the literature that it has a minimum value of zero occurring at (1, 1) [27]. The simple GA obtained this optimal value at the 51st generation in 2.35 seconds. This implies that though the Rosenbrock function has an optimal value difficult to achieve by classical optimization techniques, the precision of GA is proved by the optimal value it achieved. In addition, the simple GA has quick convergence rate. This is justified by the time the GA used to achieve the optimal value of zero. Thus, GA performs a thorough search to find near optimal value of the Rosenbrock function within a shorter time escaping most possible traps of local minima.

The exploratory nature of the GA search for best minima of the Rosenbrock function is described in Fig. 8. Though the minimum of the function is found in a valley, the average mean fitness values depict that the algorithm proved robust during the search as it had to choose from high and low fitness values at consecutive generations. For instance, after 21 generations, the mean fitness was zero but it increased to about 171 after the 47th generation. The algorithm proved robust achieving the optimal value at the end of 51 generations. These varying mean fitness values of the Rosenbrock function indicate that the function is complex.

The global minimum for an n-dimensional Rastrigin function has been found to be zero (0) at $x_i = 0$ for i =1, 2, …, n [27]. The Simple GA found the minimum value of the single-variable Rastrigin function to be zero and it occurred at zero. The optimal value of the Rastrigin function of five variables was zero. These values are indications of the accuracy of the GA to achieve optimal values within shorter time limits.

The robustness of the GA was proved by the nature of its exploration over search space whose minima are regularly distributed across the Rastrigin function (See Fig. 9). Although the algorithm started with a poor mean fitness value due to the nature of the function, it converged in subsequent generations until the 51st generation where it achieved the optimal value of the Rastrigin function (See Fig. 9). Thus, precision of the near optimal values make GAs robust and highly recommended for solving complex optimization problems.

The accuracy of GA was proved when it located the minimum of a single-variable Schwefel function at 420.9618 with an optimal value of -418.9829 in 23.02 Seconds. In the literature, the global minimum for an n-dimensional Schwefel function is -418.9829n at x_i = 420.9618 for i = 1, 2,…,n [27]. The GA further revealed the minimum of the Schwefel function of 10 variables which are presented in Table 1.

Despite the deceptive nature of the Schwefel function having consecutive best local minima distant from one another (See Fig. 6), the GA was able to produce the best minimum value as in the literature. These suggest that GAs are robust, efficient and do not compromise on time of search. In Fig. 10, it is shown that though the search was unstable in the initial generations of the search, it stabilizes at the end of the 20th generation. It converged to the optimal value from the 25th generation to the 51st generation. These prove the suitability of the GA to solve problems of such nature. These results provide enough justifications for the use of

GA for optimizing complex and multi-variable problems.

The simple GA found the minimum of a Shubert function to be -14.5080 at -0.2001. This value was reached at the 51st generation when the stopping criterion was satisfied. In general, the global minimum for an n-dimensional Shubert function is multiple of -14.5080 [27]. Further results with n = 3 is presented in Table 1.

The robustness of the GA was proved further by the search for the optimal value of complex Shubert function (Fig. 7). Though the simple GA started with a poor mean fitness value due to the nature of the function, it was able to keep the best fitness throughout other generations until a better value was found. This resulted in repeated optimal values from the 15th generation to 44th generation. This is due to the selection criteria that were implemented by the algorithm. These results indicate the ability of the GA to exhaustively search for the near optimal value of a single variable shubert function. Thus, based on the performance of the GA on the Shubert function, they show they are robust and accurate while maintaining good search over short time.

The results answer skepticism of Wang (1991) and others as to the efficiency and robustness of GAs [29,30]. Furthermore, the results prove that GAs easily escape from millions of local optima and reliably converge to a single value close to the optimum which was suggested in [31].

V. CONCLUSION

The simple GA has provided good solutions to the test functions. Except for some randomization of the initial population especially when set distant from a local optimum, the time taken by the algorithm to complete each search is small. GAs are therefore robust, works with precision and has quick convergence to the direction of the global optimum making them more desirable for complex search problems.

REFERENCES

[1]. Sivanandam SN, Deepa SN (2008) Introduction to Genetic Algorithm. Springer-Verlag Berlin Heidelberg, New York.

[2]. Francisco JS, Roger J-BW (1981) Minimization by Random Search Techniques. Mathematics of Operations Research 6(1):19-30.

[3]. Rastrigin LA (1963) The convergence of the random search method in the extremal control of a many parameter system. Automation and Remote Control 24(10):1337-1342.

[4]. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to Algorithms. MIT Press, USA.

[5]. Metropolis N, Rosenbluth AW, Rosenbluth MN, Tel AH, Teller E (1953) Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics 21(6):1087. doi:10.1063/1.1699114.

[6]. Aarts EHL, Korst J (1989) Simulated Annealing and Boltzmann Machines: a stochastic approach to combinatorial optimization and neural computing. John Wiley and Sons, USA.

[7]. Bertsimas D, Tsitsiklis J (1993) Simulated Annealing. Statistical Science 8(1): 10-15. http:// www.mit.edu/~dbertsim/papers/Optimization/Simulated\%20annealing.pdf (Last accessed: September 11, 2015)

[8]. Holland JH (1975) Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor,USA.

[9]. Goldberg D (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Professional, Reading, MA.

[10]. Naoki M, Hajime K, Yoshikazu N (2001) Adaptation to Changing Environments by Means of the Memory Based Thermodynamical Genetic Algorithm. Transactions of the Institute of Systems, Control and Information Engineers 14(1):33-41.

[11]. Leardi R (2000) Application of genetic algorithm-PLS for feature selection in spectral data sets. Journal of Chemometrics 14(5-6):643-655.

[12]. Wiendahl HP, Garlichs R (1994) Decentral Production Scheduling of Assembly Systems with Genetic Algorithm. CIRP Annals-Manufacturing Technology 43(1):389-395.

[13]. Zhang H (2008) A Genetic Algorithm for Solving RCPSP. Computer Science and Computational Technology, ISCSCT 2008. International Symposium 2: 246-249. doi:10.1109/ISCSCT.2008.255.

[14]. McCall J (2005) Genetic algorithms for modelling and optimisation. Journal of Computational and Applied Mathematics 184(1):205-222.

[15]. Karadimitriou K, Kraft DH (1996) Genetic Algorithms and the Multiple Sequence Alignment Problem in Biology. In: Proceedings of the Second Annual Molecular Biology and Biotechnology Conference, Baton Rouge, LA.

[16]. Johnson P, Vandewater L, Wilson W, Maruff P, Savage G, Graham P,Zhang P et al. (2015). Genetic algorithm with logistic regression for prediction of progression to Alzheimer's disease. BMC Bioinformatics 15(Suppl 16):S11. doi:10.1186/1471-2105-15-S16-S11.

[17]. Lin CD, Anderson-Cook CM, Hamada MS, Moore LM, Sitter RR (2015) Using Genetic Algorithms to Design Experiments:A Review. Quality and Reliability Engineering International 31(2):155-167. doi:10.1002/qre.1591.

[18]. Simpson AR, Dandy GC, Murphy LJ (1994) Genetic Algorithms Compared to Other Techniques for Pipe Optimization. Journal of Water Resources Planning and Management 120(4). doi:10.1061/(ASCE)0733-9496(1994)120:4(423).

[19]. Geisendorf S (1999) Genetic Algorithms in Resource Economic Models. Retrieved from http:// www.santafe.edu/media/workingpapers/99-08-058.pdf. (Last accessed: October 1, 2015)

[20]. Terranova OG, GarianoSL, Iaquinta P, Iovine GGR (2015) GASAKe: forecasting landslide activations by a genetic-algorithms-based hydrological model. Geosci. Model Dev. 8:1955-1978. doi:10.5194/gmd-8-1955-2015.

[21]. Torres JL, Linsangan NB (2015) Application of Genetic Algorithm for Optimization of Data in Surface Myoelectric Prosthesis for the Transradial Amputee. International Journal of Information and Education Technology 5(2):113-118. doi:10.7763/IJIET.2015.V5.486.
[22]. Leehter Y, Sethares WA (1994) Nonlinear Parameter Estimation via the genetic Algorithm. Signal Processing, IEEE Transactions 42(4):927 – 935.
[23]. Larranaga P, Poza M, Yurramendi Y, Murga RH, Kuijpers CMH. (1996) Structure learning of Bayesian Networks of genetic algorithms: a performance analysis of control parameters. IEEE Trans Patterns Anal Mach Intell 1996;18(9):912-26.
[24]. Rosenbrock HH (1960) An automatic method for finding the greatest or least value of a function. Computer Journal 3:175-184. doi:10.1093/comjnl/3.3.175.
[25]. Torn A, Zilinskas A (1989). Global Optimization. Springer-Verlag, Berlin.
[26]. Muhlenbein H, Schomisch D, Born J (1991) The Parallel Genetic Algorithm as Function Optimizer. Parallel Computing 17:619-632.
[27]. Molga M, Smutnicki C (2005). Test Functions for Optimization needs. Retrieved from http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf. (Last accessed: October 5, 2015)
[28]. Srinivas M, Patnaik LM (1994) Genetic algorithms: a survey. Computer 27(6):17-26. doi:10.1109/2.294849.
[29]. Wang QJ (1991). The Genetic Algorithm and its Application to Calibrating Conceptual Rainfall-Runoff Models. Water Resource Research 27(9):2467-2471.
[30]. Hu L, Liu J, Liang C, Ni F, Chen H (2015) A Phoenix++ Based new Genetic Algorithm Involving Mechanism of Simulated Annealing. International Journal of Distributed Sensor Networks 2015:8 pages. Doi:10.1155/2015/806708.
[31]. Jung SH (2009) Selective Mutation for Genetic Algorithms. World Academy of Science, Engineering and Technology 56:478 – 481. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.193.4022\&rep=rep1\&type=pdf. (Last accessed: September 11, 2015)