# A REVIEW OF BENCHMARKING FUNCTIONS FOR GENETIC ALGORITHMS

Vinod Goyal
GJUS&T, Hisar

Sakshi Dhingra
GJUS&T, Hisar

Anand Kumar
GJUS&T, Hisar

Dr Sanjay Singla
IET Bhaddal Technical Campus , Ropar, Punjab

**Abstract: This paper presents a review on the major benchmarking functions used for performance control of Genetic Algorithms (GAs). The difficulties of providing benchmarking of Genetic algorithms that is meaningful and logistically viable. To be meaningful the benchmarking test must give a fair comparison that is free, as far as possible, from biases that favor one style of algorithm over another. To be logistically viable it must overcome the need for pair wise comparison between all the proposed algorithms. The general behavior of two basic GAs models, the Generational Replacement Model (GRM) and the Steady State Replacement Model (SSRM) is evaluated.**

**Keywords: Genetic Algorithms Performance, Generational Replacement Model, Steady-State Replacement Model.**

## I. INTRODUCTION

Genetic algorithms (GAs) are search methods based on principles of natural selection. GAs encodes the decision variables of a search problem into finite-length strings. The strings which are candidate solutions to the search problem are referred to as chromosomes, the string are referred to as genes and the values of genes are called alleles. In contrast to traditional optimization techniques, GAs work with coding of parameters, rather than the parameters themselves. To evolve good solutions and to implement natural selection, we need a measure for good solutions from bad solutions. The measure could be an objective function that is a mathematical model, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness measure must determine a candidate solution's relative fitness, which will subsequently be used by the GA for good solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. The large population sizes lead to unnecessary expenditure of valuable computational time. Once the problem is encoded in a chromosomal manner and a fitness measure for good solutions from bad ones has been chosen, we can start to evolve solutions to the search problem using the following steps:

A. *Initialization:* The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated.

B. *Evaluation:* Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.

C. *Selection:* Selection allocates more copies of those solutions with higher fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea, including roulette-wheel selection, stochastic universal selection, ranking selection and tournament selection, some of which are described in the next section.

D. *Recombination:* Recombination combines parts of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this (some of which are discussed in the

next section), and competent performance depends on a properly designed recombination mechanism. The offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner.

E. *Mutation:* While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individual's trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution.

F. *Replacement:* The offspring population created by selection, recombination, and mutation replaces the original parental population. Many replacement techniques such as elitist replacement, generation- wise replacement and steady-state replacement methods are used in GAs.

Repeat steps B-F until a terminating condition is met.

## II. REVIEW OF BENCHMARKING FUNCTIONS

In this section various GA models and the suggested parameter sets are briefly presented by researchers in order to establish the most frequently used functions are discussed.

De Jong (1975) constructed a test of five problems in function minimization. He translated Holland's theories into practical function optimization and used two performance measures for judging the genetic algorithms. First, he defined on-line and off-line performance as an average of all costs up to the present generation and as the best cost found up to the present generation respectively. The functions suggested by De Jong have certain characteristics, which depict many of the difficulties, found in optimization problems. Following his study, a number of people suggested and tested various improvements to the basic genetic algorithm. Goldberg (1989) nicely summarizes De Jong's work.

Grefenstette(1986) used a meta-genetic algorithm to optimize the on-line and offline performance of genetic algorithms based on varying six parameters: population size Ps ,crossover probability Cp, mutation probability Mp, generation gap G, scaling window, and whether elitism is used or not.

The metagenetic algorithm used

Ps = ¼ 50, Cp ¼ 0.6, Mp ¼ 0.001, G ¼ 1.0, no scaling, and elitism.

The best genetic algorithm for on-line performance had Ps ¼ 30, Cp ¼ 0:95, Mp ¼ 0:01, G ¼1.0, scaling of the cost function and elitism.

Scwefel's (1977) test set, consisting of 62 functions that cover an enormous diversity of different topologies is surely one of the most extensive function sets.

Extensive empirical studies concerning the performance of various crossover operators have been performed by Schaffer et al., published in several papers which shed some light on the relation between performance, parameterization and configuration of Genetic Algorithms [3, 5, and 17]. Schaer et al. (1989) used discrete sets of parameter values (Ps ¼ 10; 20; 30; 50; 100; 200; Mp ¼ 0:001; 0:002; 0:005; 0:01; 0:02; 0:05; 0:10; Cp ¼ 0:05 to 0:95 in increments of 0.10 and 1 or 2 crossover points) that had a total of 8400 possible combinations.

Baeck (1991) emphasizes on two test functions (Sphere model, Rastrigin) for finding the total optimum and in the fast convergence. They had discrete sets of parameter values (probability crossover¼0.6, population size¼50, string length¼32n, mutation probability¼0.001, tournament selection and crossover two points).

Baeck et al. (1998) in formulates heuristic rules indicating that performance is decreasing both for large population size ($m > 200$) combined with large mutation probability $p_m > 0:05$ as well as for small population size ($m < 20$) combined with small mutation probability pm ¼ 0; 002. Finally, he describes several artificial test functions in Evolutionary Programming, including the sphere model, Rosenbrock's function, Ackley's function, the function after Fletcher and Powell and fractal function.

Patton et al. (1998) in explore an alternative population- based form of adaptation for evolutionary computation, termed as Guided Gaussian Mutation (GGM), which is designed specifically as a localized search operator. In testing the potential electiveness of the GGM operator, Patton selected a number of optimization test functions from the literature. Algorithms were tested with population sizes of 200 and 500. All test runs were tracked for 500 generations.

Michalewicz (1993) presented an empirical investigation of the performance of a hierarchy of evolution programs and indicated that generality grows at the expense of performance.

Salomon (1995) analyzed the benefits of small mutation rates that are commonly used and focuses on the

performance loss of typical GAs when applying a rotation to the coordinate system the theoretical analysis presented in his paper shows that most of the widely used test function have n independent parameters and that, when optimizing such functions, many GAs scale with an O(nlnn) complexity.

Pohlheim (1997) describes a number of test functions implemented for use with the Genetic and Evolutionary Algorithm Toolbox for Matlab. These functions are drawn from the literature on genetic algorithms' evolutionary strategies and global optimization.

In addition to the experiments mentioned so far, carried out by GA researchers, one can observe that there are also other suggestions as regards function sets from scientists of other relevant fields. We suggestively refer to one of the best known in the field of function optimization, by More Garbow et al. (1981) suggested a total of 34 functions which has become a landmark for many researchers since then. More Garbowin produced a relatively large collection of carefully coded test functions and designed very simple procedures for testing the reliability and robustness of unconstrained optimization software. The functions they suggested are appropriate for comparative tests of various algorithmic methods used in optimization.

Finally, another important research carried out by Ford, Riley, and Freeman involves an effort to evaluate the NAG parallel library. More specifically, they evaluate the solution improvement due the inclusion of parallelism offers to the finding of the total optimum, by providing the comparative results of the serial and parallel routines for solving the some problems. The functions they chose are also used in the work of More, Garbow and Hillstrom and are: (a) Extended Rosenbrock, (b) Watson.

Meysenburg in examine to what degree the quality of the PRNG employed affects the performance of a simple GA including eleven test functions .

Finally, there are World Wide Web pages [20, 21], which contain information on several test functions that can be used to evaluate the performance of GAs. The test suite of functions, used to evaluate the performance of GAs during the IEEE International Conference on Evolutionary Computation in 1996, is particularly interesting.

## III. SET OF BENCHMARKING FUNCTIONS

We concluded to a total of 11 optimization functions. Table I summarizes some of the widely-used test functions.

A function is multimodal if it has two or more local optima. A function of P variables is separable if it can be rewritten as a sum of P functions of just one variable.

| Function | Defination | Mutimodal | Seperable | Regular |
|---|---|---|---|---|
| Sphere | | no | yes | n/a |
| Schwefel's | $f_{Sph}(\mathbf{x}) = \sum_{i=1}^{p} x_i^2$ | no | no | n/a |
| double sum | $f_{SchDS}(\mathbf{x}) = \sum_{i=1}^{p} \left(\sum_{j=1}^{i} x_j\right)^2$ | | | |
| Rosenbrock | $x_i \in [-65.536, 65.536]$ | no | no | n/a |
| Rastrigin | $f_{Ros}(\mathbf{x}) = \sum_{i=1}^{p-1}[100(x_{i+1} - x_i^2)^2 + (x_i -$ | y es | yes | n/a |
| Schwefel | $f_{Ras}(\mathbf{x}) = 10p + \sum_{i=1}^{p}(x_i^2 - 10\cos(2\pi x_i)$ | yes | yes | n/a |
| Ackley | $f_{Sch}(\mathbf{x}) = 418.9829 \cdot p + \sum_{i=1}^{p} x_i sin\left(\sqrt{\right.$ | yes | no | yes |
| Griewangk | $f_{Ack}(\mathbf{x}) = 20 + e - 20exp\left(-0.2\sqrt{\frac{1}{p}\sum_{i=}^{p}\right.$ | yes | no | yes |
| Fletcher | $f_{Gri}(\mathbf{x}) = 1 + \sum_{i=1}^{p} \frac{x_i^2}{4000} - \prod_{i=1}^{p} cos\left(\frac{x_i}{\sqrt{i}}\right)$ | yes | no | no |
| Powell | $f_{Fle}(\mathbf{x}) = \sum_{i=1}^{p}(A_i - B_i)^2$ ) | | | |
| Langerman | $A_i = \sum_{j=1}^{p}(a_{ij} sin\alpha_j + b_{ij} cos\alpha_j$ | yes | no | no |
| | $f_{Lan}(\mathbf{x}) = -\sum_{i=1}^{m} c_i \cdot exp\left(-\frac{1}{\pi}\sum_{j=1}^{p}(x_j\right.$ | | | |

Table1

## IV. GENERAL FRAMEWORK OF METHODLOGY

The main performance metric is the efficiency of the genetic algorithm, (i.e. the ability to reach an optimum is increased the number of populations we whether the number of GA iterations required to find a solution decreased). In order to facilitate an empirical comparison of the performance of GAs, a test environment for these algorithms must be provided in the form of several objective functions f. We used the PGA Pack [8] library and we tested both population replacement models available in the field of serial GAs. The first, the generational replacement genetic algorithm (GRGA), replaces the entire population each generation and is the traditional genetic algorithm. The second variant is the steady state genetic algorithm (SSGA) in which only a few

structures are generated in each iteration (λ=1 or2) and they are inserted in the current populations Q=P(t). The criterion we used for ending the algorithm stipulates the following:
''The executions stop only if after a certain number of repetitions (in this paper 100 repetitions) the best individual is not substantially mutated.''
The population size affects both the overall performance and the efficiency of GAs. Two population replacement schemes are used. The first, the generational replacement model (GRM), replaces the entire population each generation and is the traditional genetic algorithm. The second, the steady state replacement model
(SSRM), typically replace as only a few strings each generation.
Two PNGs have been tested, i.e. (a) PGAR Uniform and (b) ISAAC (Indirection, Shift, Accumulate, Add, and Count). Each time PGAPack is run, unique sequence of random numbers is used. In the package, the initial population of individuals is created by selecting each bit uniformly at random. For each bit position of each individual in the population, a call is made to a PNG to obtain a real number in the range [0, 1].
We obtained the ISAAC PNG from the World Wide Web [22]. It is intended to be a PNG worthy of use in cryptographic applications. The claimed period for the PNG ranges from $2^{40}$ to $2^{8295}$, depending on the seed value. Then, we found the online value for each PNG and test function, and then compared this value with the true optimal value for each function. The crossover in our experiments is applied with six probability values pc 2 ½0:6; 0:95with step 0.05 [4, 8, 17], and its application or non-application is defined by a random number generator. When it is not applied, the offspring is considered as an exact copy of one of the parents with equal probability. Mutation on the other hand is applied with a 1=2l probability where l is the alphanumerical string [2, 9].

## V. CONCLUSION

An issue we examine in this paper is to what degree the performance of the PNG employed effects the performance of the GA. For functions, ISAAC PNG drove the GA to its maximum value in fewer generations. In fact, when the GA did produce an individual with optimal value, there was no statistical difference between the number of generations required to produce the individual when the GA was driven by a good PNG (ISAAC) and when it was driven by PGARUniform. When optimizing multimodal functions, a PNG has even more severe consequences. Therefore, the experiments described above are important in that they identify approximately optimal parameter settings for the four performance measures considered.

## VI. REFERENCES

[1.] Baeck, T. (1991). ''Optimization by Means of Genetic Algorithms''. In Internationales Wissenschaftliches Kolloquium (pp. 1–8). Germany: Technische University''.

[2.] Baeck T. (1996). Evolutionary Algorithms in Theory and Practice, New York: Oxford University Press, pp. 138–159.

[3.] Caruana, R. A., Eshelman, L. J. and Schaer, J.D. (1989). ''Representation and hidden bias II: Eliminating defining length bias in genetic search via shue crossover''. In N. S. Sridharan, editor, Proceedings of the 11th International Joint Conference on Artificial Intelligence, pp.750–755. Morgan Kaufmann Publishers, San Mateo, CA.

[4.] De Jong, K. A. (1975). ''An analysis of the behaviour of a class of genetic adaptive systems'', University of Michigan, Ann Arbor. (University Microfilms No. 76-9381).

[5.] Eshelman, L. J., Caruana, R. A., and Schaer, J.D. (1989). ''Biases in the crossover landscape'', In Proceedings of the 3rd International Conference on Genetic Algorithms and Their Applications, Morgan Kaufmann Publishers, San Mateo, CA, pp. 10–19.

[6.] Ford, R. W. and Riley, G. D. (1997). ''The development of Parallel Optimization Routines for the NAG Parallel library'', High Performance Software for Nonlinear Optimisation Conference, Naples.

[7.] Goldberg, D. E. (1989). ''Genetic algorithms in search, optimization and machine learning'', New York: Addison-Wesley, pp.40–45.

[8.] Grefenstette, J. J. (1986). ''Optimization of control parameters for genetic algorithms'', in Sage, A. P. (Ed.), IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-16 (pp. 122–128). New York: IEEE GENETIC BENCHMARKING FUNCTIONS415.

[9.] Levine, D. (1996). ''Users Guide to the PGAPack Parallel Genetic Algorithm Library'', Argonne National

Laboratory, Mathematics and Computer Science Division, pp. 19–28.

[10.] Meysenburg, M. M. (1997). ‘‘The eect of the quality of Pseudo-Random Number Generator Quality on the Performance of a Simple Algorithm’’, College of Graduate Studies, University of Idaho.

[11.] Michalewicz, Z. (1993). ‘‘A hierarchy of evolution programs: An experimental study’’. Evolutionary Computation, 1(1) pp. 51–76.

[12.] Vinod Goyal, “A Review of Benchmarking Functions for Genetic Algorithms”, National Conference on Advanced Computing and Communication Technology,ACCT-2010,pp- 363-337.

[13.] More, J. J., Garbow, B. S., and Hillstrom, K.E. (1981). ‘‘Testing Unconstrained Optimization Software, {ACM} Transactions on Mathematical Software, vol 7, pp. 17–41.

[14.] Patton, A. L. , Dexter, T., Goodman, E. D. and Punch, W. F. (1998). ‘‘On the Application of Cohort-Driven Operators to Continuous Optimization Problems using Evolutionary Computation’’, in Lecture Notes in Computer Science, vol 1447, pp. 671–682, Springer - Verlag Inc.

[15.] Pohlheim, H. (1997). ‘‘GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB’’, http://www.systemtechnik.tuilmenau.de/~pohlheim/GA_Toolbox.

[16.] Salomon. R. (1995). ‘‘Reevaluating GeneticAlgorithm Performance under Coordinate Rotation of Benchmark Functions’’, BioSystems vol. 39, pp. 263–278, Elsevier Science.

[17.] Salomon, R. (1997). ‘‘Improving the Performance of Genetic Algorithms through Derandomization.’’ In G. Pomberger (Ed.), Software - Concepts and Tools, vol 18, pp. 175–184, Springer-Verlag, Berlin.

[18.] Schwefel, J. D., Caruna, R. A., Eschelman, L.J. and Das, R. (1989). ‘‘A study of control parameters aecting online performance of genetic algorithms for function optimization’’, in J. D. Schaer (Ed.), Proceedings of the 3rd International Conference on Genetic Algorithms and Their Applications, pp. 61–68, Morgan Kauman Publishers, San Mateo, CA.

[19.] Schwefel, II. P. (1977). Numerische optimierung von Computer-Modellen mittels derEvolutionsstrategie, In Interdisciplinary Systems Research vol. 26, pp. 319–354, Birkha¨user, Basel

[20] To¨ rn, A. and Zilinskas, A. (1991). Global Optimization, volume 350 of Lecture Notes in Computer Science, Springer, Berlin.

[21] First international contest on evolutionary optimization. http://iridia.ulb.ac.be/langerman/ ICEO.html, 1997.

[22] Optimization test functions - L. Lazauskas.http://www.maths.adelaide. edu.au/Applied.llazausk/ alife/realfopt.htm, 1997.

[23] Random number generators, http://ourworld.compuserve.com/home pages/bob_jenkins/;randomnu.htm, 1997. 416 ,J. G. DIGALAKISJ. G. Digalakis And K. G. Margaritis, “On Benchmarking Functions For Genetic Algorithms“, Intern. J. Computer Math., Vol. 79(4), pp. 403–416, 2002.

[24] Kumara Sastry, David Goldberg, “Genetic Algorithms”.