How Random Generator Quality Impacts Genetic Algorithm Performance

Mark M. Meysenburg, Dan Hoelting, Duane McElvain
Computer Science Dept.
Doane College
Crete, NE 68333

James A. Foster
Computer Science Department
University of Idaho
Moscow, ID USA 83844

Abstract

It has been shown that pseudo-random number generator (PRNG) choice can affect simple genetic algorithm (GA) performance. However, these performance impacts are non-intuitive; PRNGs of poor quality can drive GAs to superior performance, for certain problems. The same PRNGs cause worse performance for other problems. In this paper we present a plausible explanation for this phenomenon: PRNGs of poor quality cause higher Vose discrepancy values than do higher quality PRNGs. Higher Vose discrepancy values could then be manifest as GA performance differences, as GA populations move toward fixed points of the Vose heuristic far away from the expectation.

1 INTRODUCTION

Several researchers have examined the impact of pseudo-random number generator (PRNG) choice on genetic algorithm (GA) performance. Meysenburg and Foster [Meysenburg, 1997, Meysenburg and Foster, 1997] examined several PRNGs, using the Knuth [Knuth, 1997] and Marsaglia's Diehard [Marsaglia, 1993] empirical test suites. They used the PRNGs to drive a simple GA, applied to a collection of several well-known GA test functions. Using a relatively coarse-grained statistical measure, they found no statistical evidence that PRNG quality affected GA performance.

In a second study Meysenburg and Foster [Meysenburg and Foster, 1999b] developed a set of specific, empirical PRNG quality tests tailored to the way a simple GA uses randomness. They used a similar set of PRNGs and the same set of GA test functions as in the previous work. They found, however, that there was no correlation between good performance on the PRNG tests and good performance by the GA. In the second study, however, a finer statistical measure was used that did reveal an interesting phenomenon.

One of the PRNGs used was a version of the Java language Random generator, limited to a period of 1000 numbers. With such a limited period, this PRNG (rand1k) failed the PRNG tests miserably. However, there was evidence that rand1k affected GA performance. It would be reasonable to assume that worse PRNG quality would cause worse GA performance, but this was not the case.

On several of the GA test functions, rand1k caused the GA to perform better than other, much better, PRNGs. On other functions, rand1k caused the GA to perform worse than the other PRNGs. In summary, Meysenburg and Foster's second study found that there was evidence that PRNG choice could impact GA performance, although in non-intuitive ways. Similar results have been noted for genetic programming (GP) systems [Meysenburg and Foster, 1999a, Daida et al., 1997, Daida et al., 1999].

In summary, the research to date on this subject shows that PRNG choice can impact GA (or GP) performance. However, the research shows no direct correlation between improved PRNG quality and improved GA performance; in fact, better PRNGs can in some cases cause worse GA performance. No one has yet been able to explain why PRNG choice can alter GA performance in this manner.

2 GA THEORY

Vose [Vose, 1999] has developed a general mathematical theory describing the behavior of simple GAs. Vose calls the search space explored by the GA Ω. If the size of Ω is n, then GA populations can be represented as vectors in n-space. These population vectors are elements of a set that Vose terms the simplex:

$$\Lambda = \left\{ \langle x_0, \ldots, x_{n-1} \rangle : \mathbf{1}^T x = 1, x_j \geq 0 \right\}. \qquad (1)$$

Elements of the simplex are column vectors of size n, where each component of the vector is non-negative, and all components of the vector sum to one. A vector $p \in \Lambda$ represents a population as follows: component p_j is the percentage of the whole of the j^{th} element of Ω in the GA population.

A GA is defined in terms of a transition rule $\tau : \Lambda \rightarrow \Lambda$, describing how a GA population evolves over time. Given an initial population vector p, the next generation would be $\tau(p)$; the following generation would be $\tau(\tau(p)) = \tau^2(p)$; and so on. Unfortunately, we are unable to say with certainty what $\tau(p)$ would be, because GAs are stochastic algorithms.

To deal with the stochastic nature of GAs, Vose introduces another function $\mathcal{G} : \Lambda \rightarrow \Lambda$, called the heuristic function. For a population vector p, the result of $\mathcal{G}(p)$ is another vector $q \in \Lambda$. q is then used as a sampling distribution to produce the next generation. The j^{th} component of q is the probability that the j^{th} element of Ω is selected to be a member of the next generation. The various operators of the GA (selection, crossover, and mutation, for example) are implemented in the particular heuristic $\mathcal{G}$ chosen. The GA population is moved forward by applying $\mathcal{G}$ to the initial population p, and using the resulting sampling distribution to create the next population. The process repeats until termination criteria are met.

Given an initial population vector p, repeated applications of the heuristic $\mathcal{G}$ produce a path through n-space. This is the expected path the GA population should follow during a run. Fixed points of $\mathcal{G}$ correspond to situations where the GA converges. The actual path followed by a GA, of course, will vary to a certain degree from the expectation, due to the stochastic nature of the process.

Vose has developed a formula for determining how far away from the expected path a particular GA population vector is.

For population vector p, the probability that the next population vector is q is shown in Figure 1. In the formula, the summations are only done for indexes where $q_j > 0$, and r is the number of individuals in the GA population.

In Figure 1, the term

$$\sum q_j \log \frac{q_j}{\mathcal{G}(p)_j} \qquad (2)$$

is called the discrepancy of q with respect to the expectation $\mathcal{G}(p)$. The discrepancy is a measure of how far the actual next population vector, q, is from the expected next population vector, $\mathcal{G}(p)$. It is a measure of the distance between expectation and reality.

Our current research has shown that Vose's theory can be used to explain the non-intuitive GA behavior observed in previous studies [Meysenburg, 1997, Meysenburg and Foster, 1997, Meysenburg and Foster, 1999b]. Our hypothesis is that a PRNG of quality poor enough to drive the GA population far from the path predicted by Vose theory, would cause the GA to perform differently than a GA driven by a PRNG of higher quality. We hypothesized that a PRNG like rand1k would cause higher Vose discrepancy values for successive GA populations than a high quality PRNG like the Mersenne Twister [Matsumoto and Nishimura, 1998] would. Then rand1k might drive the GA populations into the basins of attraction of different Vose heuristic fixed points than the Mersenne Twister would; this would account for GA performance differences.

3 EXPERIMENT DESIGN

In order to test our hypothesis, we first collected 42 GA test problems suitable for Vose discrepancy statistic calculation. Since the complexity of the discrepancy measure is $O(3^l)$, for chromosome length l, the statistic can only be efficiently computed for chromosomes of approximate length 20 or less. Our test functions were created as part of an undergraduate research project. The functions are based on several different classes of problems drawn from the literature, adapted to our chromosome length restrictions. The functions have chromosome lengths ranging from eight to 20. Our GA test problems are briefly summarized in Table 1. More detailed descriptions of each of the problems may be found on the World Wide Web at the following URL: `http://ist.doane.edu/meysenburg/cooperstuff/index.html` . This page describes each test problem, as well as the parameters (crossover and mutation rates, population size, etc.) used for each run.

Next, we ran a simple GA (of the type described by Vose [Vose, 1999]) on each of the 42 GA test problems. We repeated the runs for each of 14 different PRNGs, ranging in quality from rand1k to the Mersenne Twister. Finally, to reduce the likelihood of anomalies caused by poor seed value selection, we repeated each of our runs for 32 different PRNG seed values. For each problem / seed value combination, we initialized the GA population identically, and then used the PRNG under test for the rest of the GA run. In this way, each of the runs for a problem / seed value pair started at the same point in Ω. The seed values and initial populations were constructed using the truly random source at `www.random.org` .

We then used the Mann-Whitney non-parametric statistical test to determine if PRNG choice caused performance differences in our GA runs. We compared average population fitness on a generation by generation basis in a manner similar to Meysenburg and Foster's second study [Meysenburg and Foster, 1999b].

Finally, we calculated the Vose discrepancy statistic between each generation of each GA run. These calculations are complete for every GA test function where $l < 20$, and are still under way for the problems where $l = 20$. We used the Wilcoxson non-parametric statistical test to determine if discrepancy values caused by the rand1k PRNG were greater than those caused by the other PRNGs.

4 RESULTS

In our experiments, we again found that PRNG choice impacts GA performance. Our statistical measures here did not indicate if a PRNG caused better or worse GA performance than the other PRNGs; the measures only detected that a difference (in either direction) existed. Of all our GA runs, we found that the rand1k PRNG caused performance differences in 68% of the cases. None of our other PRNGs caused consistent performance differences across the 42 GA test functions.

Having confirmed that rand1k causes unexpected GA performance, we next tried to determine if the poor quality of rand1k caused higher Vose discrepancy values than our other PRNGs. For the GA test functions we have had time to calculate Vose discrepancy statistics for, this is indeed the case. Representative results for three of our shorter-length GA test functions are shown in Tables 2, 3, and 4.

The DC_19 GA test function has chromosome length $l = 12$. The function is an instance of CNF-SAT, for 12 variables, 300 clauses, and five variables per clause. The bits of the chromosome determine the values of each variable.

The DC_37 and DC_41 GA test functions have chromosome length $l = 8$. These functions are a modified version of the emergency-unit placement problem described by Haupt and Haupt [Haupt and Haupt, 1998]. In this case, an emergency response building must be placed on a city map, represented as a 16 by 16 grid, with a river cutting across the map at row seven. A bridge is placed over the river to allow vehicles to cross the river. For the DC_37 function, the bridge is in column one of row seven, while in the DC_41 function, the bridge is in column seven of row seven.

In the figures, the letter 'W' represents a case where the row-label PRNG caused statistically higher Vose discrepancy values compared to the column-label PRNG. The figures show that, for these GA test functions, rand1k causes higher discrepancy values than any of our other PRNGs. Other PRNGs cause sporadic Vose discrepancy differences, but rand1k causes higher Vose discrepancies compared to all other PRNGs, in all of the GA test functions we have computed the statistics on so far. We speculate that the sporadic Vose discrepancy differences of other PRNGs are caused by the small population size of our GA runs; Vose theory says that higher discrepancy values are likely in small population GAs.

It is interesting that the infamous RANDU PRNG [Knuth, 1997], which scores as badly as rand1k in the Diehard suite of PRNG quality tests, does not impact the GA in the same way rand1k does. In particular, RANDU never caused GA performance differences in our runs (while rand1k did 68% of the time), and neither did RANDU cause consistently higher discrepancy values than the other PRNGs (while rand1k did). Therefore, it seems that the Diehard suite is not predictive for GA use. We have developed a GA-specific empirical test of PRNG quality (described in a poster presented at this conference [Meysenburg et al., 2002]) which eliminates this false positive problem. Our new test, tailored to the specific GA parameters of our test functions, gives poor scores to rand1k but normal scores for RANDU.

In summary, for the GA functions we have been able to examine to date, rand1k does cause higher Vose discrepancy values than other, higher quality PRNGs.

5 CONCLUSIONS AND FURTHER WORK

We have shown that poor PRNG quality does correlate with abnormally high Vose discrepancy values. We feel that this correlation explains why a poor quality PRNG, such as rand1k, can cause improved or degraded GA performance, compared to other PRNGs. High enough discrepancy values could cause the GA to enter the basins of attraction of unexpected fixed points of the Vose heuristic; this would be manifest as GA performance differences.

In order to further bolster our confidence in our hypothesis, we are continuing Vose discrepancy calculations on our larger GA test functions. As the results become available, we will determine if the correlation between poor PRNG quality and high Vose discrepancy values continues. In addition, we would like to determine the fixed points of the Vose heuristic for our GA test functions, in order to confirm that rand1k drives GA populations to fixed points different than other PRNGs do.

Acknowledgments

This work is supported by the Doane College Cooper Undergraduate Research Program, the Initiative for Bioinformatics and Evolutionary STudies (IBEST) at the University of Idaho; by NIH NCRR grant 1P20RR016454-01; and by NIH NCRR grant NIH NCRR 1P20RR016448-01; and by NSF grant NSF EPS 809935.

$$
\begin{aligned}
&\Pr\left\{\tau\left(p\right)=q\right\}\\
&\quad = r!\prod\frac{\left(\mathcal{G}(p)_j\right)^{rq_j}}{\left(rq_j\right)!}\\
&\quad = exp\left(-r\sum q_j\log\frac{q_j}{\mathcal{G}\left(p\right)_j}-\sum\left(\log\sqrt{2\pi rq_j}+\frac{1}{12rq_j+\Theta\left(rq_j\right)}\right)+O\left(\log r\right)\right)
\end{aligned}
$$

Figure 1: Vose equation for probability that population q came from population p.

Function	**Name**	**Length**	**Function**	**Name**	**Length**
DC_01	Rastrigin's Function	20	DC_22	Ackley's Trap Function	20
DC_02	Michalewicz's Function	16	DC_23	Ackley's 1-Max Function	20
DC_03	Whitley's Function	20	DC_24	Ackley's Mix Function	20
DC_04	Rana's Function	20	DC_25	Ackley's Plateaus Function	20
DC_05	Schwefel's Function	20	DC_26	Hoelting's Projectile	16
DC_06	Griewangk's Function	20	DC_27	Koza's Cart-Pole	20
DC_07	Schaffer's Function	20	DC_28	New Light's Bug Bomb	16
DC_08	McElvain's Fibonacci	16	DC_29	Haupt's 4-letter Word Guesser	20
DC_09	Shaffer's Function	20	DC_30	Koza's Cart-Pole II	20
DC_10	Keane's Bump Function	20	DC_31	Koza's Cart-Pole III	20
DC_11	Shopping Cart Packing	18	DC_32	Koza's Cart-Pole IV	20
DC_12	Function F9	20	DC_33	6-city TSP	18
DC_13	Schubert's Function	20	DC_34	Max Clique	16
DC_14	16-200-4 CNF-SAT	16	DC_35	6-city TSP II	18
DC_15	16-50-3 CNF-SAT	16	DC_36	6-city TSP III	18
DC_16	20-80-3 CNF-SAT	20	DC_37	Haupt's ERU Location	8
DC_17	15-5-5 CNF-SAT	15	DC_38	Haupt's ERU Location II	8
DC_18	20-80-3 CNF-SAT II	20	DC_39	Real Topology Hill-Climber	9
DC_19	20-300-5 CNF-SAT	20	DC_40	Binary-to-Gray Circuit	17
DC_20	Ackley's 2-Max Function	20	DC_41	Haupt's ERU Location III	8
DC_21	Ackley's Porcupine	20	DC_42	Meysenburg's DFA	18

Table 1: Doane College GA Test Suite functions

	add	fsr	mersenne	mother	pm	rand	rand1k	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w											
fsr														
mersenne														
mother														
pm		w	w								w			w
rand		w	w											
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu		w												
shlec		w												
shpm														
shsub														
sub														
tauss		w												
tgfsr														

Table 2: Vose discrepancy results for DC_19

	add	fsr	mersenne	mother	pm	rand	rand1k	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w		w	w		w	w	w	w	w		
fsr														
mersenne									w	w				
mother		w	w		w			w	w	w	w	w		
pm									w					
rand									w					
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu														
shlec														
shpm														
shsub									w					
sub									w	w				
tauss		w	w		w	w		w	w	w	w	w		
tgfsr		w			w			w	w	w	w			

Table 3: Vose discrepancy results for DC_37

	add	fsr	mersenne	mother	pm	rand	rand1k	randu	shlec	shpm	shsub	sub	tauss	tgfsr
add		w	w											
fsr														
mersenne														
mother														
pm		w	w								w			w
rand		w	w											
rand1k	w	w	w	w	w	w		w	w	w	w	w	w	w
randu		w												
shlec		w												
shpm														
shsub														
sub														
tauss		w												
tgfsr														

Table 4: Vose discrepancy results for DC_41

References

[Daida et al., 1997] Daida, J., Ross, S., McClain, J., Ampy, D., and Holczer, M. (1997). Challenges with verification, repeatability, and meaningful comparisons in genetic programming. In Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., and Riolo, R. L., editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 64–69, Stanford University, CA, USA. Morgan Kaufmann.

[Daida et al., 1999] Daida, J. M., Ampy, D. S., Raatanasavetavadhana, M., Li, H., and Chaudhri, O. A. (1999). Challenges with verification, repeatability, and meaningful comparison in genetic programming: Gibson's conundrum. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, FL, USA. Morgan Kaufmann.

[Haupt and Haupt, 1998] Haupt, S. and Haupt, R. (1998). *Practical Genetic Algorithms.* John Wiley and Sons.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming*, volume 2. Addison Wesley, third edition.

[Marsaglia, 1993] Marsaglia, G. (1993). Monkey tests for random number generators. *Computers & Mathematics with Applications*, 9:1–10.

[Matsumoto and Nishimura, 1998] Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3 – 30.

[Meysenburg, 1997] Meysenburg, M. M. (1997). The effect of pseudo-random number generator quality on the performance of a simple genetic algorithm. Master's thesis, University of Idaho.

[Meysenburg and Foster, 1997] Meysenburg, M. M. and Foster, J. A. (1997). The quality of pseudorandom number generators and simple genetic algorithm performance. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 276 – 281. Morgan Kaufmann.

[Meysenburg and Foster, 1999a] Meysenburg, M. M. and Foster, J. A. (1999a). Random generator quality and gp performance. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann.

[Meysenburg and Foster, 1999b] Meysenburg, M. M. and Foster, J. A. (1999b). Randomness and ga performance, revisited. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann.

[Meysenburg et al., 2002] Meysenburg, M. M., Hoelting, D., McElvain, D., and Foster, J. A. (2002). A genetic algorithm-specific test of random generator quality. In *GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann.

[Vose, 1999] Vose, M. D. (1999). *The Simple Genetic Algorithm.* MIT Press.