Testing PRNG's for use in a GA

Clyde Meli
Assistant Lecturer
CIS Department
University of Malta
Tel: 2340-2509

cmeli @ cis.um.edu.mt

ABSTRACT

In this paper, the results of some tests including the Chi-Square and the Diehard Battery tests of randomness on some pseudo random number generators are given. The generators were chosen for use in a Genetic Algorithm (GA) package called GAGENES, written in object-oriented C++. The influence of PRNG's on a GA is discussed briefly.

Categories and Subject Descriptors

I.2.m [**Artificial Intelligence**]: Miscellaneous – *PRNG, testing, genetic algorithms*..

General Terms

Algorithms, Measurement.

Keywords

ga,gagenes,prng,testing,c++,object-oriented

1. INTRODUCTION

Four pseudo random generators were tested using a Chi-Square test [2] and the Diehard tests [3]. Meysenburg [5] by empirical testing showed that a good PRNG may not necessarily give significantly better GA performance. Meysenburg et al [6] determined that there is statistical evidence that certain PRNGs can provide improved GA perfomance. Cantu-Paz [1] showed that in GAs as in other fields, it is preferable to use the best PRNG available especially when initialising the GA population. With this in mind, four PRNG's were tested as follows: 1) a Chi-Square test was implemented ; 2) a 12M data file was generated for every PRNG for testing by Marsaglia's Diehard program. All PRNG's were seeded by the system time. The results are presented in this paper. The generators tested were: the PRNG supplied with GNU GCC C++ 3.4.5 / libstdc++'s linear congruential generator (LCG), Strousoup's PRNG implementation (an LCG) in [7], Knuth's Lehman LCG [2] (this was implemented in C++) and Jasper Bedaux's C++ port of the Mersenne Twister [4]. The latter was expected to be the best PRNG from those tested.

George Marsaglia's **DIEHARD** suite of statistical tests consist of fifteen tests: Birthday Spacings, Overlapping Permutations, Ranks of 31x31 and 32x32 matrices, Ranks of 6x8 matrices, Monkey Tests on 20-bit Words, Monkey Tests OPSO, OQSO and DNA, Count the 1's in a stream of bytes, Count the 1's in specific bytes, Parking Lot, Minimum Distance, 3D Spheres, Squeeze, Overlapping Sums, Runs and Craps. More information about the suite may be found at http://stat.fsu.edu/~geo/diehard.html

To run the test, Marsaglia's Windows binary was used and it was given a twelve megabyte file of random numbers generated using GAGENE's random number test suite (which called the four PRNG's under test). All tests were run on an Athon FX-55 processor computer running XP. The test suite also compiles and runs under Linux and other Posix style OS's.

2. CHI-SQUARE RESULTS

Each simple frequency Chi-Square test was repeated 100000 times and the mean chi value was calculated. The range of output values was subdivided into a fixed number of ranges (101) of equal size. A large number of random numbers were generated using the PRNG being tested, each number ranging between 0 and 1 (inclusive). 1000000 numbers were generated and the number of values falling in each interval were counted. As the intervals are of equal size, it is expected that approximately the same number of values should fall in each interval (around 9900). The chi-square statistic is evaluated based on the expected and actual counts for each interval, an average chi-square value based on 100000 repetitions is evaluated and compared with the critical value for a chi-square random variable with 100 degrees of freedom and a probability of 0.95.

Chi-Square Test 1 (GNU C++ generator)
Total chi value (Averaged over 100000 trials)=90.1802
succeeded
Chi-Square Test 2 (Mersenne generator)
Total chi value (Averaged over 100000 trials)=100.013
succeeded
Chi-Square Test 3 (Stroustroup generator)
Total chi value (Averaged over 100000 trials)=-1.59487e+06
failed
random Chi-Square Test 4 (Knuth generator)
Total chi value (Averaged over 100000 trials)=13768.5
failed

When run on a Linux machine (Pentium 4), the results were:

random Chi-Square Test 1 (GNU C++ generator)
Total chi value (Averaged over 100000 trials)=99.2055
succeeded
random Chi-Square Test 2 (Mersenne generator)
Total chi value (Averaged over 100000 trials)=100.263
succeeded
Chi-Square Test 3 (Stroustroup generator)
Total chi value (Averaged over 100000 trials)=-1.59487e+06
failed
random Chi-Square Test 4 (Knuth generator)
Total chi value (Averaged over 100000 trials)=13941.9
failed

3. DIEHARD RESULTS

The following table shows the p-values produced by the Diehard suite of tests on the GCC C++ generator (resulting log file is rcpp.log).

Table 1. GCC C++ Diehard Test Results Part 1

Birthday Spacing	**Bin. Rank 31x31**		
.384929	.660257		
.718312	**Bin. Rank 32x32**		
.319485	.602642		
.945639	**Bin. Rank 6x8**		
.579601	.927291	.739037	.631103
.489100	.870957	.222028	.475929
.843946	.193213	.093370	.947564
.498052	.073664	.162025	**.220698 KSTEST**
.360428	.967210	.338048	
.510440 KSTEST	.514890	.496957	
OPERM5	.682539	.192736	
.171280	.456832	.455710	
.323761	.046014	.478321	
	.589094	.105389	
	.602133	.445554	

Table 2. GCC C++ Diehard Test Results Part 2

Bitstream	**OPSO**	**OQSO**	**DNA**	**CNT 1sB**	**3D SPH**
.69693	.0282	.0000	1.0000	.774920	90260
.38483	.0372	.0000	.0000	.561840	72882
.76172	.0005	.0246	1.0000	.579544	03619
.28882	.1765	.1693	.0000	.426853	78345
.80420	.0013	.6130	.0008	.486733	69484
.70665	.0751	.0018	.0896	.669792	24076
.70585	.1400	.1971	.2083	.104575	69320
.92183	.0161	.0000	1.0000	.504758	84426
.23998	.0320	.0000	1.0000	.581833	89207
.76244	.1874	.0000	1.0000	.156646	19813
.22081	.0000	.0440	1.0000	.868356	10472
.65583	.0140	.2067	.0000	.554558	70579
.41734	.1332	.7550	.0002	.885662	38517
.44672	.0177	.5508	.0229	.725794	43728
.49503	.0034	.0110	.2383	.062972	05089
.79833	.7007	.0000	.1200	.969570	74042
.11519	.0517	.0000	1.0000	.990735	62511
.25474	.0005	.0000	.0000	.547370	96707
.90331	.0164	.0162	1.0000	.961884	63341
.52764	.0218	.0024	.0000	.934765	55377
	.0011	.9759	.0148	.805098	**.531503**
CNT 1s	.1362	.1162	.2832	.094834	KSTEST
.321054	.0478	.0000	.0001	.345356	
.463177		.0000	1.0000	.297482	**OSUM**
		.0000	1.0000	.619431	.157359
CDPARK		.0000	.0000		.840688
.205562		.0026	1.0000		.147470
.180558		.0001	.0000		.891995
.590298			.0371		.294137
.590298		**SQEEZE**	.0080		.603960
.108811		.317070	.0200		.471933
.246694					.036775
.291865		**RUNS**	**CRAPS WINS**		.573609
.934075		.079560	.765831		.552787
.500000		.964089	**CRAPS THROWS**		**.075029 KSTEST**
.136563		.488655	.104740		
.759000 KSTEST	**MIN .DIST**	.069318			
	.965807				

The next table shows the p-values produced by the Diehard suite of tests on the Mersenne generator (resulting log file is rmersey.log)..

Table 3. Mersenne Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
.76244	.9321	.7635	.5513	.906931	.35889
.98490	.4844	.5117	.5963	.103090	.21381
.93460	.0627	.7421	.4098	.268602	.62479
.82600	.9473	.6168	.5883	.666844	.01906
.84785	.3303	.9354	.8368	.849311	.68057
.41734	.4149	.3208	.5963	.021068	.85892
.56097	.1626	.9930	.2109	.851932	.71178
.58480	.0843	.6711	.1870	.203728	.83036
.00862	.8611	.8745	.6468	.941688	.08710
.11702	.9620	.8302	.6100	.497766	.27134
.55359	.5989	.2638	.6134	.577195	.06129
.26306	.8765	.1426	.6695	.277723	.74412
.30581	.7797	.2841	.7547	.992178	.00882
.41006	.6069	.8961	.7260	.845530	.98379
.57384	.4029	.6513	.9778	.466425	.35548
.29282	.3673	.1110	.2964	.147168	.88393
.29282	.0966	.1433	.8609	.482845	.30497
.87659	.9399	.8215	.9981	.255032	.26642
.24143	.9543	.5468	.9500	.705604	.63249
.07200	.4830	.4363	.4259	.294328	.95606
	.1362	.9033	.8888	.496514	**.291919**
CNT 1s	.8900	.5360	.8331	.151208	**KSTEST**
.668328	.8291	.2627	.3245	.729378	
.044188		.0437	.3213	.961371	**OSUM**
		.3605	.6302	.981730	.523352
CDPARK		.9201	.6357		.955821
.753306		.0837	.4926		.000946
.554479		.7789	.2903		.613807
.708135			.2704		.111499
.246694		**SQEEZE**	.0371		.564409
.192812		.508527	.7808		.929658
.914635					.326266
.000951		**RUNS**	**CRAPS WINS**		.427841
.625377		.675138	.854552		.442707
.536382		.678659	**CRAPS THROWS**		**.386011 KSTEST**
.015932	**MIN**	.129089	..040774		
.637284 KSTEST	**DIST**	.429987			
	.536019				

Table 4. Mersenne Diehard Test Results Part 2

Birthday Spacing	Bin. Rank 31x31		
.126317	.432115		
.954330	**Bin. Rank 32x32**		
.283559	.956891		
.743912	**Bin. Rank 6x8**		
.553217	.250621	.417829	.097548
.581776	.083185	.539246	.354312
.585145	.328546	.369348	.053574
.357520	.699241	.156892	**.239078 KSTEST**
.508808	.767513	.919956	
.196512 KSTEST	.887186	.479579	
OPERM5	.230224	.375273	
.148375	.541184	.718831	
.271223	.034657	.881512	
	.227694	.915275	
	.654906	.411544	

The next table shows the p-values produced by the Diehard suite of tests on the Stroustroup generator (resulting log file is rmersey.log).

Table 5. Stroustroup Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.5548	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
	1.00000	1.00000	1.0000	1.000000	**1.000000 KSTEST**
CNT 1s	1.00000	1.00000	1.0000	1.000000	**KSTEST**
1.000000	1.00000	1.00000	1.0000	1.000000	
1.000000		1.00000	1.0000	1.000000	**OSUM**
		1.00000	.0000	1.000000	1.000000
CDPARK		1.00000	.0000		1.000000
.000000		1.00000	.0000		1.000000
.000000		1.00000	1.0000		1.000000
.000000			1.0000		1.000000
.000000		**SQEEZE**	1.0000		1.000000
.000000		RTE	1.0000		1.000000
.000000					1.000000
.000000		**RUNS**	**CRAPS WINS**		1.000000
.000000		.481974	RTE		1.000000
.000000		.816563	**CRAPS THROWS**		**1.000000 KSTEST**
.000000	**MIN**	.595713	RTE		
1.000000 KSTEST	**DIST**	.510214			
	1.000000				

RTE means the test produced a Runtime Error and had to be skipped.

Table 6. Stroustroup Diehard Test Results Part 2

Birthday Spacing	Bin. Rank 31x31		
1.000000	1.000000		
1.000000	**Bin. Rank 32x32**		
1.000000	1.000000		
1.000000	**Bin. Rank 6x8**		
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	**1.000000 KSTEST**
1.000000	1.000000	1.000000	
1.00000 KSTEST	1.000000	1.000000	
OPERM5	1.000000	1.000000	
.676957	1.000000	1.000000	
.024215	1.000000	1.000000	
	1.000000	1.000000	
	1.000000	1.000000	

The next table shows the p-values produced by the Diehard suite of tests on the Knuth generator (resulting log file is rknut.log)..

Table 7. Knuth Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97455
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97528
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
	1.0000	1.0000	1.0000	1.000000	**1.000000**
	1.0000	1.0000	1.0000	1.000000	KSTEST
	1.0000	1.0000	1.0000	1.000000	
		1.0000	1.0000	1.000000	
		1.0000	1.0000	1.000000	
		1.0000	1.0000		
		1.0000	1.0000		
		1.0000	1.0000		
			1.0000		
			1.0000		
			1.0000		

CNT 1s
1.000000
1.000000

CDPARK
.000000
.000000
.000000
.000000
.000000
.000000
.000000
.000000
.000000
.000000
1.000000 KSTEST

OSUM
.759934
.980955
.978755
.962520
.946317
.940524
.981306
.919689
.951057
.965671
1.000000 KSTEST

SQEEZE
1.000000

RUNS
1.000000
1.000000
1.000000
1.000000

CRAPS WINS
.295951

CRAPS THROWS
.********

MIN DIST
1.000000

Table 8. Knuth Diehard Test Results Part 2

Birthday Spacing
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000 KSTEST

OPERM5
1.000000
1.000000

Bin. Rank 31x31
1.000000

Bin. Rank 32x32
1.000000

Bin. Rank 6x8		
1.000000	1.000000	1.000000
1.000000	1.000000	1.000000
1.000000	1.000000	1.000000
1.000000	1.000000	**1.000000 KSTEST**
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	

4. DIEHARD ANALYSIS

A note provided in the Diehard log files says

> NOTE: Most of the tests in DIEHARD return a p-value,
> should be uniform on [0,1) if the input file contain
> independent random bits. Those p-values are obtain
> p=F(X), where F is the assumed distribution of the s
> random variable X---often normal. But that assumed F
> an asymptotic approximation, for which the fit will
> in the tails. Thus you should not be surprised with
> occasional p-values near 0 or 1, such as .0012 or .9
> When a bit stream really FAILS BIG, you will get p's
> 1 to six or more places. By all means, do not, as a
> Statistician might, think that a p < .025 or p> .975
> that the RNG has "failed the test at the .05 level".
> p's happen among the hundreds that DIEHARD produces,
> with good RNG's. So keep in mind that " p happens".

For the C++ PRNG, the smallest p-value was 0 and the largest was 1.000. This PRNG passed most Diehard tests with some failures in OPSO, OQSO and mainly DNA tests.

The Mersenne Twister PRNG passed all the Diehard tests successfully. The smallest p-value was .000951 and the largest p-value was .9930. The only areas with p-values over .975 were OPSO, OQSO, DNA, 3D Spheres and Count-The-1's for specific bytes. Areas with p-values under 0.025 were OSUM, CDPARK and 3D Spheres.

Diehard gave a runtime error when running the Squeeze and Craps tests on the Stroustroup PRNG. The only tests which it passed were OPERM5 and RUNS. P-values ranged from 0 to 1.

The Knuth linear congruential generator passed the 3D Spheres test (except for the KSTEST (Kolmogorov-Smirnov test) and Craps Wins (for Craps Throws Diehard outputted just '*****' instead of a p-value, it is not clear whether this is some overflow or underflow).

5. CONCLUSION

The Mersenne Twister PRNG clearly emerged as the best PRNG from the ones tested and the one most likely to be suitable for use during the critical initalisation phase of a GA. For the normal GA run after initialisation, however the standard C++ PRNG may be used. Further research may give a more clear picture on the effects of a bad PRNG (one that fails the tests) with regards to the running of the GA (mainly post-initialisation), for some particular problem and whether this depends on population size.

A copy of the random binary files generated, the Diehard logfiles and other sources will be made available on the GAGENES website at http://www.gagenes.com.

6. REFERENCES

[1] Cantu-Paz, E. On Random Numbers and the Performance of Genetic Algorithms.

[2] Knuth, D.E. *The Art of Computer Programming*. Addison-Wesley, 1997.

[3] Marsaglia, G., Tsang, Wai Wan Some difficult-to-pass tests of randomness. *Journal of Statistics Software*, 7(3), 2003.

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

[5] Meysenburg, M.M. The Effect of Pseudo-Random Number Generator Quality on the Performance of a Simple Genetic Algorithm *Computer Science*, University of Idaho, 1997, 82.

[6] Meysenburg, M.M., Foster, James, Randomness and GA Performance, Revisited. in *Proceedings of the Genetic and Evolutionary Computation Conference*, (San Francisco, California, 1999), Morgan Kauffman Publishers, 425-432.

[7] Stroustroup, B. *The C++ Programming Language Special Edition*. Addison-Wesley, 2005.