Good Pedagogical Random Number Generators

J. Stanley Warford
Department of Computer Science
Pepperdine University
Malibu, CA 90265

warford@pepvax.bitnet
warford@pepvax.pepperdine.edu

Abstract A *CACM* paper by Park and Miller [6] advocated a standard for random number generators based on the Lehmer generator [5] and criticised a number of computer science textbooks for presenting bad random number generators. This paper advocates the proposed standard and presents a set of generators based on theoretically sound principles that are also useful for microcomputer implementation and classroom presentations at the introductory level.

Introduction

The article by Park and Miller [6] describes the proliferation of bad random number generators in subroutine libraries and in textbooks. They make a strong case for standardizing on the Lehmer generator [5] of the form

$$z_{n+1} = az_n \bmod m$$

where z_n is the nth random number in the sequence, m is the modulus, a large prime integer, and a is the multiplier, an integer in the range [2..m–1]. The pseudorandom integers produced by such a generator lie in the range [1..m–1] and must repeat with a cycle length of at most m–1. Although shorter cycle lengths are possible depending on the choice of a and m, generators that achieve the maximum cycle length are known as full-period generators.

Among the bad generators Park and Miller describe are those that are based on the above algorithm but are not full-period, those that are full-period but that are not sufficiently random, and those that are based on other algorithms. A common example of an inferior algorithm is the sequence

$$z_{n+1} = (az_n + c) \bmod m$$

where the additive term c satisfies $c \bmod m \neq 0$. The effect of c is to allow $z = 0$ in the sequence, so that a full-period sequence has a cycle length of m instead of $m-1$. Implementations of generators based on this algorithm usually have m equal to a power of 2 so the mod operation can be done efficiently with a binary shift. Although it seems intuitively that the presence of c should "mix up" the sequence, it turns out that such a term (when m is a power of 2) will *always* introduce a highly nonrandom behavior in the sequence. The least significant bit will cycle with a period of 2, the next significant bit with a period of 4, and so on. One result is that the terms of the sequence will alternate perfectly between odd and even.

Other bad generators include algorithms that depend on the overflow arithmetic of a specific processor and hence are not portable in high-level languages, and algorithms that depend on the uncontrolled precision of real computations such as operations that use truncated approximations of π. To show how widespread the bad generators are, Park and Miller cite no fewer than 17 computer science textbooks with faulty generators!

Park and Miller propose a standard generator as that Lehmer generator with $a = 7^5 = 16807$ and $m = 2^{31}-1$ (a Mersenne prime). This generator satisfies their three criteria for a good generator: (1) a full-period sequence, (2) a sufficiently random sequence, and (3)

implemented efficiently with 32-bit arithmetic. This generator has passed extensive statistical tests and is known to be one of the best.

One good feature of the proposed minimal standard is that m is in excess of 2 billion and is therefore good for serious simulations. There is an application, however, that requires a cycle length that is much less than the cycle length of the proposed standard, namely the problem of teaching the concept of pseudorandom numbers. This application requires a small value of m so examples that illustrate the generator will involve computations that are manageable on common hand-held calculators.

Not wanting to present an inferior generator as in the texts cited by Park and Miller, I searched for values of a and m in the Lehmer generator that would be suitable for teaching purposes [9]. The three criteria of Park and Miller were adopted with (3) modified to be implemented efficiently with 16-bit arithmetic. The resulting random number generators are simple, easy to teach, and are based on a theoretically sound algorithm. They have the additional advantage of being useful for very small simulations, such as interactive games of chance.

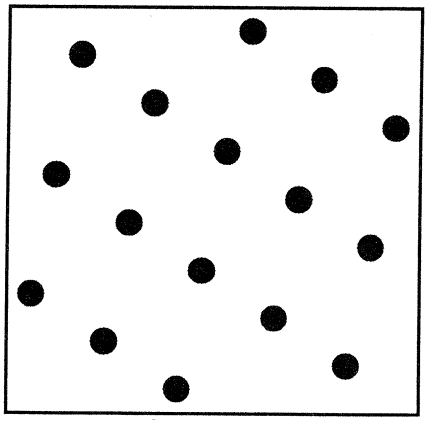
The next section describes the spectral test, a test for randomness used for criterion (2) in the search for good values of a and m. The following section gives the results of the search in the form of a table of the best values of a and m over a range of m that is suitable for 16-bit integer arithmetic common on microcomputers.

The Spectral Test

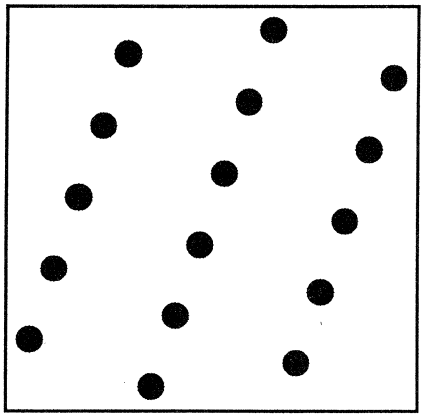
Criteria (1) and (3) together provided the first filter for the exhaustive search. Restricting m to a prime number prevents z from ever becoming zero, but does not guarantee a full-period sequence. There are 560 primes between 10 and 4096. For each prime, m, the multipliers, a, between 2 and min (m–1, trunc (32767/m)) were determined that produced a full-period sequence. This restriction guarantees that the intermediate multiplication, az, will not overflow a 16-bit integer system. A few primes in this range have no full-period multipliers, but most have several.

The next problem was to determine the randomness test for criterion (2). Although many chi-square tests are possible for testing the randomness of a sequence, I selected the spectral test as described in Knuth [3]. According to Knuth, "not only do all good generators pass this test, all generators now known to be bad actually *fail* it. Thus it is by far the most powerful test known ...".

The spectral test is based on the position of points in Euclidian space whose co-ordinates are successive subsequences of the generated numbers. As an illustration of the spectral test consider the $m = 17$, $a = 5$ generator. This full-period generator produces the sequence 1, 5, 8, 6, 13, 14, 2, 10, 16, 12, 9, 11, 4, 3, 15, 7, 1, after which the cycle repeats. The two-



(a) $m = 17$, $a = 5$



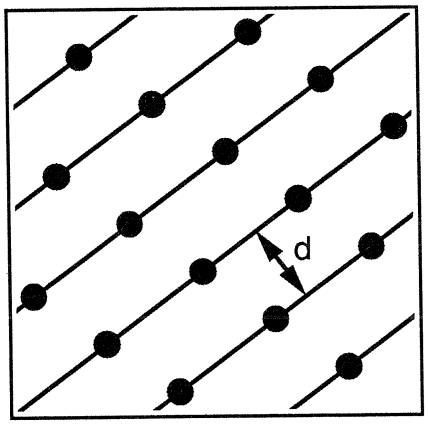
(b) $m = 17$, $a = 3$

Figure 1. Two full-period multipliers for the modulus 17.

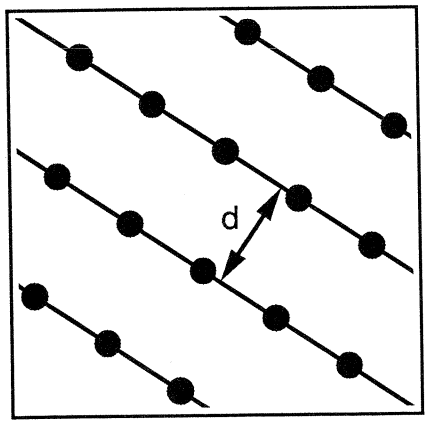
dimensional spectral test analyzes the points in the plane whose coordinates are (1, 5), (5, 8), (8, 6), (6, 13), ... , (7, 1). Each co-ordinate is then normalized by dividing by m. The result is a set of points in the unit square as shown in Figure 1(a). For comparison, the points for the m = 17, a = 3 generator are shown in Figure 1(b). Its sequence is 1, 3, 9, 10, 13, 5, 15, 11, 16, 14, 8, 7, 4, 12, 2, 6, 1.

It would appear that the points in Figure 1(a) are in some sense more "spread out" than those in Figure 1(b), and those in Figure 1(b) have more of a regular pattern. The spectral test quantifies this notion and determines that the a = 3 sequence is, in fact, less random than the a = 5 sequence.

The test constructs a family of parallel, equally-spaced lines that cover all the points in the unit square. The length d is the distance between adjacent lines. Figure 2(a) shows one family of parallel lines that cover the points of the m = 17, a = 5 generator. This family has lines spaced a distance d = 1/5 = 0.200 apart. Figure 2(b) shows another family covering the same set of points, but whose line spacing is $d = 1/\sqrt{13}$ = 0.277. Many other coverings are possible. For example, you could cover the points with a family of 16 vertical (or horizontal) lines spaced 1/17 = 0.059 apart since the coordinates are all multiples of 1/17.



(a) d = 1/5 = 0.200.



(b) $d = 1/\sqrt{13}$ = 0.277.

Figure 2. Two families of parallel straight lines that cover the points from the m = 17, a = 5 generator.

It turns out that the family of lines in Figure 2(b) is the one with the largest inter-line spacing for the m = 17, a = 5 generator. It is readily apparent from Figure 1(b) that the points for the m = 17, a = 3 generator can be covered with several families of lines, but the family with the maximum interline distance is the one with three lines in the northeast direction. This family has d = $1/\sqrt{10}$ = 0.316.

The spectral test defines $1/\nu_2$ to be the maximum distance between lines, taken over all families of parallel lines that cover the points in two dimensions. ν_2 is therefore the two-dimensional accuracy of the sequence. The a = 5 sequence is better than the a = 3 sequence because its $\nu_2 = \sqrt{13}$ is greater than the $\nu_2 = \sqrt{10}$ of the a = 3 sequence.

The two-dimensional spectral test analyzes adjacent pairs of pseudorandom integers. The three-dimensional test analyzes adjacent triples. For example, the co-ordinates in 3-space for the points from the m = 17, a = 5 generator are (1, 5, 8), (5, 8, 6), (8, 6, 13), (6, 13, 14), ... , (7, 1, 5). These 16 points can be covered by several families of equally spaced parallel planes in the unit cube. ν_3 is the reciprocal of the maximum inter-plane distance over all familes of planes.

The idea generalizes to arbitrarily high dimensions. ν_t is the reciprocal of the maximum inter-hyperplane distance taken over all parallel hyperplane families that cover the points in t-dimensional Euclidean space. Moving to the next higher dimension in the test corresponds to increasing the length of the analyzed subsequences by 1. Standard practice is to compute ν_t for values of t in the range $2 \le t \le 6$,

corresponding to analyzing all subsequences of length 6 and less.

Results

Knuth's [3] figure of merit, μ_t, which is proportional to v_t^t/m has the advantage of being relatively independent of m. This figure of merit was computed for all six dimensions of all full-period Lehmer generators with m between 10 and 4096. For a given dimension, a larger μ_t, corresponds to a better generator. The smallest μ_t, of the six dimensions is therefore a "worst case" value. The figure of merit for a given generator is the minimum μ_t over all six dimensions, $\mu_{\min}$. The results are shown in the table below, which defines the best generator in a subrange of m to be the one with the maximum $\mu_{\min}$. Values of v_t^2 are shown to indicate the absolute accuracy of each generator.

Knuth considers the multiplier to pass the spectral test if μ_t is 0.1 or more for $2 \le t \le 6$ and to be exceptional if μ_t is greater than 1. Although these generators all pass the test, the caveat is that their small cycle lengths make them good only for illustrative purposes or for extremely small applications like simulating some rolls of a pair of dice, shuffling a deck of cards, or taking a short random walk around the block.

Conclusions

Park and Miller's paper elicited several letters in a subsequent issue of CACM [1, 2, 8]. They mention in their reply [7] that they had received requests for random number generators on systems with only 16-bit integer arithmetic. Their advice was to get a better system or use the 16-bit generator presented by L'Ecuyer [4]. In undergraduate teaching environments, however, the first option is not always feasible. Furthermore, L'Ecuyer's generator is unnecessarily complicated for pedagogical purposes. Besides, why teach a complex algorithm when a simple one is being advocated as a standard?

As computer science educators, we should continually be on the alert so that the algorithms we present to our introductory students are the ones that are the best known. The generators presented here are offered in that spirit. They will not only allow you to present the standard Lehmer algorithm but provide you with specific good small values for m and a as well.

References

[1] Abrahams, P.W., Technical Correspondence. *Commun. ACM* 32, 8 (Aug. 1989), 1021-1022.

[2] Edgeman, R.L., Technical Correspondence. *Commun. ACM* 32, 8 (Aug. 1989), 1020-1021.

[3] Knuth, D.E. *The Art of Computer Programming*. Vol. 2 *Seminumerical Algorithms*. 2nd Ed. Addison-Wesley, Reading, Mass., 1981.

[4] L'Ecuyer, P. Efficient and portable combined random number generators. *Commun. ACM* 31, 6 (June 1988), 742-749, 774.

[5] Lehmer, D.H. Mathematical methods in large-scale computing units. *Annu. Comput. Lab. Harvard Univ.* 26 (1951), 141-146.

Best generator in range	has modulus m of	and multiplier a of	with $\mu_{\min}$ equal to	v_2^2	v_3^2	v_4^2	v_5^2	v_6^2
10-20	17	5	2.40	13	5	3	3	3
20-50	37	5	2.21	26	9	6	3	3
50-100	83	58	2.64	73	14	7	6	4
100-200	139	101	2.76	122	21	9	7	5
200-500	467	24	3.24	482	61	18	10	7
500-1000	797	30	3.01	898	69	23	13	9
1000-2000	1013	28	2.43	785	90	25	15	11
2000-4096	2027	15	0.35	226	85	26	21	11

[6] Park, S.K., and Miller, K.W. Random number generators: Good ones are hard to find. *Commun. ACM* 31, 10 (Oct. 1988), 1192-1201.

[7] Park, S.K., and Miller, Technical Correspondence. *Commun. ACM* 32, 8 (Aug. 1989), 1023-1024.

[8] Sand, F.M., Technical Correspondence. *Commun. ACM* 32, 8 (Aug. 1989), 1022-1023.

[9] Warford, J. S. *Computer Science.* D.C. Heath and Company, Lexington, Mass., 1991.