Computer Physics Communications 184 (2013) 1889–1897

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Finding the best portable congruential random number generators

Fatin Sezgin [a,*], Tevfik Metin Sezgin [b,1]

[a] Bilkent University, Ankara, Turkey
[b] Koc University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 14 December 2012
Received in revised form
3 March 2013
Accepted 12 March 2013
Available online 22 March 2013

Keywords:
Lattice structure
Linear congruential generator
Portability
Random number
Spectral test

ABSTRACT

Linear congruential random number generators must have large moduli to attain maximum periods, but this creates integer overflow during calculations. Several methods have been suggested to remedy this problem while obtaining portability. Approximate factoring is the most common method in portable implementations, but there is no systematic technique for finding appropriate multipliers and an exhaustive search is prohibitively expensive. We offer a very efficient method for finding all portable multipliers of any given modulus value. Letting $M = AB+C$, the multiplier A gives a portable result if $B-C$ is positive. If it is negative, the portable multiplier can be defined as $A = \lfloor M/B \rfloor$. We also suggest a method for discovering the most fertile search region for spectral top-quality multipliers in a two-dimensional space. The method is extremely promising for best generator searches in very large moduli: 64-bit sizes and above. As an application to an important and challenging problem, we examined the prime modulus $2^{63} - 25$, suitable for 64-bit register size, and determined 12 high quality portable generators successfully passing stringent spectral and empirical tests.

1. Introduction

Random numbers are essential tools in many applications, such as simulation, education, cryptography, the arts, numerical analysis, computer programming, VLSI testing, recreation and sampling. Besides some physical and tabular sources, there are several deterministic computational techniques to produce random sequences of data, including congruential, shift register, lagged Fibonacci, inverse and cellular automata generators. Linear congruential generators have attracted the attention of many researchers because they are efficient, easy to implement and their theory is well documented.

Mixed linear congruential generators produce a sequence of integers $\{X_i\}$ defined by the recursion

$$X_{n+1} = AX_n + c (\mathrm{mod}\ M) \tag{1}$$

with appropriately defined integer constants A, c, M and an initial value X_0. The special case of $c = 0$ has particular significance and is called a multiplicative congruential generator. The maximum possible period of these generators, $M - 1$, is limited by the maximum integer size available in the computer and compiler. For example, 32-bit computers can process integers up to $2^{31} - 1$. This Mersenne prime is a popular modulus for multiplicative generators for 32-bit arithmetic. Since the product AX_n can produce very large values, several methods have been suggested in the literature to prevent integer overflow.

This study considers the portable implementation, relying on the so-called approximate factoring method by using the constants A, B and C, satisfying the expression $M = AB + C$ such that $B > C$. In the article, the term "portable" is used specifically to refer to approximately factorable generators.

In Section 2 we offer a systematic search method for efficiently determining all multipliers suitable for approximate factoring. Section 3 gives a fruitful search interval for the best-quality congruential generators with respect to a spectral test criterion. In Section 4, we present application examples. In Section 5, we apply the method to the modulus $M = 2^{63} - 25$, suitable for 64-bit computers, and report 12 very good multipliers. In the last section, we summarize the results and suggest some future applications.

2. Implementations for portable generators

Consider the multiplicative congruential generator $X_{n+1} = 48271X_n (\mathrm{mod}\ (2^{31} - 1))$. The calculations required by this generator may produce integers up to 1.473×2^{46} and cause an integer overflow in 32-bit arithmetic. There are several studies proposing and reviewing techniques for the correct implementation of random number generators, such as [1–8].

* Corresponding author. Tel.: +90 312 290 5010; fax: +90 312 266 4607.
E-mail addresses: fatin@bilkent.edu.tr (F. Sezgin), mtsezgin@ku.edu.tr (T.M. Sezgin).
[1] Tel.: +90 212 338 1540; fax: +90 212 338 1548.

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.03.013

2.1. Portable implementation by approximate factoring

Among many implementation methods, approximate factoring has become highly popular for its effectiveness and simplicity. This method relies on the decomposition of the modulus M as

$$M = AB + C \tag{2}$$

where

$$B = \left\lfloor \frac{M}{A} \right\rfloor \tag{3}$$

and $C = M - AB$, subject to the condition $B > C$, and with $\lfloor\cdot\rfloor$ denoting the floor function. It was first used by Wichmann and Hill [9] in an informal remark. Sezgin [7] and Park and Miller [10] independently proved the validity of the method. According to this implementation, the code $X_{n+1} = AX_n (\mathrm{mod}\ M)$ can be written as:

$K = int(X_n/B)$
$X_{n+1} = A * (X_n(\mathrm{mod}\ B)) - C * K$
if $(X_{n+1}.lt.0)X_{n+1} = X_{n+1} + M$.

For example, since $2147483647 = 48271 * 44488 + 3399$, the generator

$$X_{n+1} = 48271X_n (\mathrm{mod}\ (2^{31} - 1))$$

can be coded as:

k = ix/44488
ix = 48271 * mod(ix, 44488) − 3399 * k
if(ix.lt.0)ix = ix + 2147483647.

We state the following theorem for the condition of portability:

Theorem 1. *For a modulus M, the multiplier A is portable if and only if* $\left\lfloor \frac{M}{A} \right\rfloor > \frac{M}{A+1}$.

Proof. Assume that A is portable. Then in (2) we have $B > C$. This implies that $B > M - AB$. Inserting the value of B from (3) we get

$$\left\lfloor \frac{M}{A} \right\rfloor > M - A\left\lfloor \frac{M}{A} \right\rfloor$$

or

$$\left\lfloor \frac{M}{A} \right\rfloor (A + 1) > M.$$

Dividing both sides by $A + 1$ gives the asserted result. The opposite is also true. Assume that $\left\lfloor \frac{M}{A} \right\rfloor > \frac{M}{A+1}$. Then the operations in the reverse direction give $\left\lfloor \frac{M}{A} \right\rfloor (A + 1) > M$ and this implies $BA + B > M = AB + C$, giving $B > C$. □

Corollary 1. *All multipliers* $A \leq \sqrt{M}$ *are portable.*

Proof. It is easy to see that $A = \sqrt{M}$ is portable. Because in this case the above theorem is satisfied:

$$\left\lfloor \frac{M}{\sqrt{M}} \right\rfloor > \frac{M}{\sqrt{M} + 1}.$$

On the other hand, when $A < \sqrt{M}$ we will have $B \geq A$. But since $A > C$, we get $B > C$. □

Corollary 2. *There is no portable multiplier larger than* $\left\lfloor \frac{M}{2} \right\rfloor$.

Table 1
Finding multipliers suitable for approximate factoring in modulus $M = 103$.

A	B	C	$B - C$	$BX >$	X is	Portable $A =$
13	7	12	−5	5	1	$14 = 13 + 1$
15	6	13	−7	7	2	$17 = 15 + 2$
18	5	13	−8	8	2	$20 = 18 + 2$
21	4	19	−15	15	4	$25 = 21 + 4$
26	3	25	−22	22	8	$34 = 26 + 8$
35	2	33	−31	31	16	$51 = 35 + 16$

Proof. $\left\lfloor \frac{M}{2} \right\rfloor$ is portable, because in this case we will get the B value

$$B = \left\lfloor \frac{M}{A} \right\rfloor = \left\lfloor \frac{M}{\left\lfloor \frac{M}{2} \right\rfloor} \right\rfloor = 2.$$

Therefore the C value will be $C = M - AB = M - 2\left\lfloor \frac{M}{2} \right\rfloor = 1$ or 0 depending on the M being odd or even. We also see that $A = \left\lfloor \frac{M}{2} \right\rfloor + k$ is not portable for positive integers k. Because if $k \geq 1$, we will get $B = 1$, and $B > C$ implies $C = 0$. But this is not the case since we get $C = M - AB = M - A \geq B$. □

We note that we always have $C = M(\mathrm{mod}\ A) < A$, but most of the time $C > B$. The portability condition is met by few multipliers. For example, there are only 92679 multipliers satisfying this condition for $M = 2^{31} - 1$ and this number constitutes 0.0043% of all multiplier candidates. Hence, an exhaustive search for portable generators is fruitless; some general rules must be determined for an efficient search algorithm. In this respect, we present the following remarks:

1. All multipliers $\leq \sqrt{M}$ can be used for approximate factoring.
2. The last portable multiplier by approximate factoring is $\lfloor M/2 \rfloor$.
3. Above $\sqrt{M}$, in some cases we may have $B - C < 0$. If that is the case, we need an X value such that $XB + (B - C) > 0$. In order to get a portable multiplier, this X value must be added to A.

Example. Let us take $M = 103$, a small modulus, for demonstrative purposes. For values up to $A = 10$, all multipliers can be used in approximate factoring. We observe that 11 and 12 are also satisfactory. But $A = 13$ gives $B = 7$ and $C = 12$, causing a negative $B - C$ value: $B - C = -5$. We must have $7X > 5$, getting $X = 1$. When this occurs, $13 + 1 = 14$ can be implemented by approximate factoring. Following the same method, we get five more approximate factoring generators, as summarized in Table 1.

The following theorem explains the method for finding the next portable multiplier when $B < C$:

Theorem 2. *When a multiplier* $\sqrt{M} < A < M/2$ *gives* $B < C$, *the next portable generator can be obtained by using the multiplier* $\left\lfloor \frac{M}{B} \right\rfloor$.

Proof. From (2) we get $B - C = B - M + AB$. If this value is negative, we need to add an X value to A to get a positive $B - C$. It means that we will have $BX > -(B - C) = M - B - AB$ or $BX > M - (A + 1)B$. The X value will be

$$X > \left\lfloor \frac{M - (A + 1)B}{B} \right\rfloor.$$

We can take the smallest possible integer X as

$$X = \left\lfloor \frac{M - (A + 1)B}{B} \right\rfloor + 1.$$

This can be simplified as:

$$X = \left\lfloor \frac{M}{B} \right\rfloor - (A + 1) + 1$$
$$= \left\lfloor \frac{M}{B} \right\rfloor - A.$$

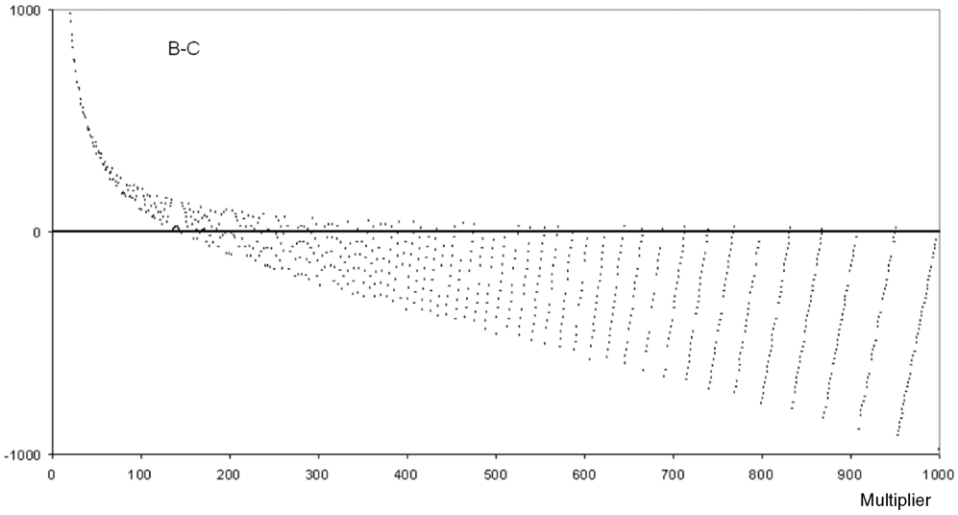

Fig. 1. The distribution of $B - C$ values for the modulus $M = 19997$ until $A = 1000$.

Therefore, when $B - C < 0$, the next portable multiplier must be taken simply as $A + X = \left\lfloor \frac{M}{B} \right\rfloor$. □

For example: With $M = 103, A = 35$; $X = \left\lfloor \frac{103-36\times2}{2} \right\rfloor + 1 = \left\lfloor \frac{31}{2} \right\rfloor + 1 = 16$; therefore $35 + 16 = 51$ is portable. Since $B = \left\lfloor \frac{M}{A} \right\rfloor = \left\lfloor \frac{103}{35} \right\rfloor = 2$, the same multiplier can be obtained simply as $\left\lfloor \frac{M}{B} \right\rfloor = \left\lfloor \frac{103}{2} \right\rfloor = 51$.

A FORTRAN program for finding portable multipliers is presented in Listing 1. It receives modulus values from the keyboard and stops when 0 is given.

```
      integer a, b
1     print*, "M=? Give 0 to stop"
      read*, M
      if (m.eq.0) stop
      maxa=m/2
      a=sqrt(float(m))
      print*, "All multipliers less than or equal to ",a," are
      portable"
      print*, "Other portable multipliers are:"
2     a=a+1
      if (a.gt.maxa) goto 1
      b=m/a
      if(b.gt.m-a*b) then
      print*, a
      else
      a=m/b
      print*, a
      end if
      goto 2
      end
```

Listing 1. The FORTRAN program finding portable multipliers.

2.2. The distribution of $B - C$

$B - C$ can be expressed as a function of A

$$B - C = \left\lfloor \frac{M}{A} \right\rfloor - M + A \left\lfloor \frac{M}{A} \right\rfloor.$$

It has a positive value until $A \leq \sqrt{M}$. After this limit negative values start to be seen. For large values of A, $B-C$ values are located on straight lines with a slope $\left\lfloor \frac{M}{A} \right\rfloor$ and the proportion of negative points increases. The distribution of $B - C$ is demonstrated for modulus $M = 19997$ in Fig. 1. The $B - C$ values are positive for all multipliers $A \leq \sqrt{M} = 141$. The first negative value is seen at $A = 146$, and as the multiplier gets larger, the proportion of negative values increases, and the values points cluster on straight lines. For large A, at each line there is only one positive value; this situation explains the inefficiency of the exhaustive search for portable multipliers. The figure shows multipliers until $A = 1000$. The remaining multipliers have the same pattern of lines, only with one positive point. Until $\sqrt{M}$, all multipliers are portable for all M. After the square root, the number of portable multipliers declines. The cumulative distribution of portable multipliers is depicted in Fig. 2 for $M = 2^{31} - 1$. Each point represents 10 000 multiplier candidates. An inspection of the figure shows that 50% of the portables are below $A = 50000$. Only 10% of the portables are above $A = 230000$. The percentage of portable multipliers above $A = 500000$ is lower than 5%. In the light of the discussion above, an exhaustive search for large portable multipliers is impractical for two reasons:

1. These multipliers are very scarce and searching for them is a waste of time.
2. As will be seen in Section 3, the spectral quality of large portable multipliers is extremely poor.

3. Selecting good multipliers

After finding multipliers suitable for implementation by approximate factoring, the next step is to determine the full-period condition and the quality of these multipliers. Random number generators must be subjected to several theoretical and empirical tests before they can be used in serious applications. In this respect, the spectral test is a very reliable theoretical tool to distinguish between bad and good congruential generators. This test is explained in detail by Knuth [11]. Letting $0 < A < M$ and A be relatively prime to M, the test determines the values of

$$\nu_t = \min \left\{ \sqrt{\sum_{i=1}^{t} s_i^2} \right\} \tag{4}$$

for $2 \leq t \leq T$, given that

$$\sum_{i=1}^{t} s_i A^{i-1} \equiv 0 \pmod{M} \tag{5}$$

where s_i are integers $0 \leq s_i < M$ and $(s_1, \ldots, s_t) \neq (0, \ldots, 0)$.

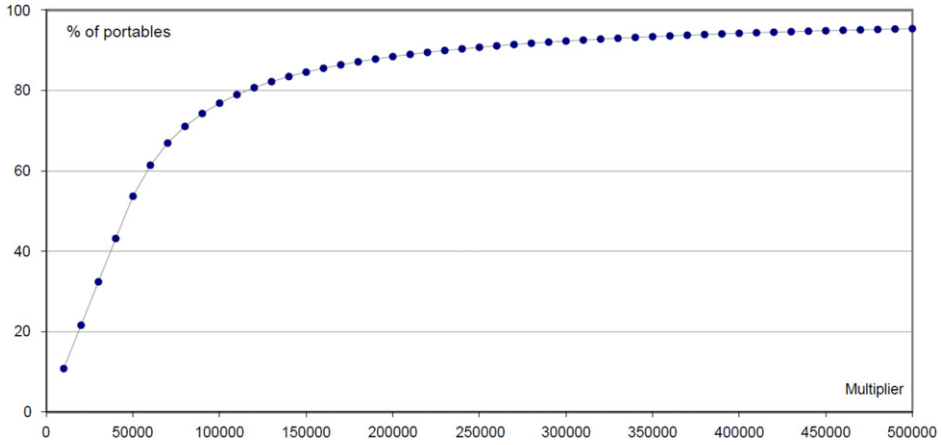

Fig. 2. Cumulative distribution of portable multiplier proportions for modulus $M = 2^{31} - 1$.

Table 2
The upper and lower limits of the best multipliers for the modulus $M = 2^{31} - 1$.

Percentile	Lower limit of A	Upper limit of A	Lower limit of S_2
Best 20%	$42374 = 0.914\sqrt{M}$	$71370 = 1.540\sqrt{M}$	0.851
Best 10%	$44963 = 0.970\sqrt{M}$	$67157 = 1.449\sqrt{M}$	0.903
Best 5%	$46207 = 0.997\sqrt{M}$	$65197 = 1.407\sqrt{M}$	0.928
Best 1%	$48008 = 1.036\sqrt{M}$	$53790 = 1.161\sqrt{M}$	0.964

Letting $d_t = 1/\nu_t$ represent the maximum distance between adjacent hyper-planes determined by the points of the lattice in a t-dimensional space, and d_t^* represent the lower bound of this distance, a figure of merit called the normalized spectral test value is defined as

$$S_t = d_t^*/d_t. \tag{6}$$

For our portable multipliers, if A is small, the identity (5) can be defined as $A - A = 0$ (Mod M), implying $s_1 = 1$ and $s_2 = -A$. By (4) these values will produce the following minimum ν_2 value:

$$\nu_2 = \min\left\{\sqrt{\sum_{i=1}^{2} {s_i}^2}\right\} = \sqrt{1 + A^2}.$$

Since d_2^* is the constant $1/M^{1/2}(4/3)^{1/4}$, from (6), the normalized spectral value for $t = 2$ will be:

$$S_2 = \frac{\sqrt{1 + A^2}}{\sqrt{M}(4/3)^{1/4}}.$$

This is approximately $0.93A/\sqrt{M}$ and explains the steady increase in the quality of small multipliers when they approach the square root of the modulus. S_2 will reach its maximum value $S_2 = 1$ when $A = \sqrt{M}/0.93 = 1.075\sqrt{M}$. When the multiplier exceeds $\sqrt{M}$ by a large margin, this situation produces very small ν_2 values. Because a very large A creates a smaller B value, and since C is even smaller, taking the identity (5) as $AB + C = M$ (Mod M) $= 0$, the generator produces a very small ${\nu_2}^2 = B^2 + C^2$ value. The search for the best two-dimensional lattice structure can be restricted to a narrow interval. Investigating the distribution of S_2 values in Fig. 5 for portable multipliers $30000 < A < 68000$ in a generator with $M = 2^{31} - 1$, we can notice that the peak $S_2 = 0.9985$ is reached at $A = 49883 = 1.076\sqrt{M}$. Sezgin [12] presented percentiles of normalized spectral test for dimensions $t = 2, \ldots, 8$. These percentiles remain unchanged across modulus values. Table 2 presents the intervals for the best multiplier percentile limits for $t = 2$ in $M = 2^{31} - 1$.

4. Some numerical examples

In Section 2, the distribution of $B-C$ was described, and a graph was presented in Fig. 1 for a small modulus. Here, in Fig. 3, we demonstrate the pattern of large multipliers for a larger modulus, $M = 2^{31} - 1$, from $A = 300000$ to $A = 310000$. There are only 231 non-negative values (portable multipliers) in this range. The proportion of portables is 2.31%. As the multipliers increase, this proportion will fall sharply.

For large modulus values like $M = 2^{48}$, 2^{63} or 2^{128}, only partial searches are conducted in the random number literature, because, as mentioned in Section 3, conducting an exhaustive search for good portable generators is a tedious task. Even with restrictions on the search, it requires a great amount of computer time. Beginning in 2006 and ending in 2009, Tang and Chang [13] conducted an exhaustive search for good 64-bit congruential random number generators under some restrictions on an IBM computer running Linux and compiled with gcc-02. After this effort, they found only 35 portable multipliers having a spectral performance value above $S_T > 0.760$. Fig. 4 depicts $A/\sqrt{M}$ values for these best generators. It is interesting to observe that these generators could have been found in a very short time, because all the multipliers are extremely close to $\sqrt{M}$.

We conducted a search for $M = 2^{31} - 1$. Fig. 5 shows the distribution of S_2 values for portable multipliers $30000 < A < 68000$. Actually, the graph starts as a straight line from $A = 2$ and continues near to $\sqrt{M}$ in this manner. In later parts, there is an interesting fractal structure and a steady decrease of the spectral value with increasing A. The remaining part is depicted in Fig. 6 until $A = 2000000$ exhibiting the sharp decline in the spectral value.

An example for a larger modulus is depicted in Fig. 7: $M = 4503599090499601$ in a narrow interval near the square root for $S_2 > 0.91$ values. Results agree with our remark in Section 3. The search gave the spectral maximum A as 72179138, which is also $1.076\sqrt{M}$.

As proven in [12,14], the squared figure of merit defined in (4), ${\nu_t}^2$, changes as a polynomial of degree $2(t - 1)$ and this causes very chaotic behavior in higher dimensions. Therefore, in determining the best multipliers, one should conduct the search in two-dimensional spectral values first. Fig. 8 shows S_2, S_3 and S_4 values for a 63-bit modulus: $M = 9223372012704246017$. Although its square root, $A = 3037000496$, is very good for a two-dimensional space and gives $S_2 = 0.931$ there, it causes a very small spectral value in dimensions above two. This fact can be observed from the graphs of S_3 and S_4 in Fig. 8. The situation is similar for higher dimensions. But fortunately, the bad lattice range is very short. In the worst case, within a range of length about 2900,

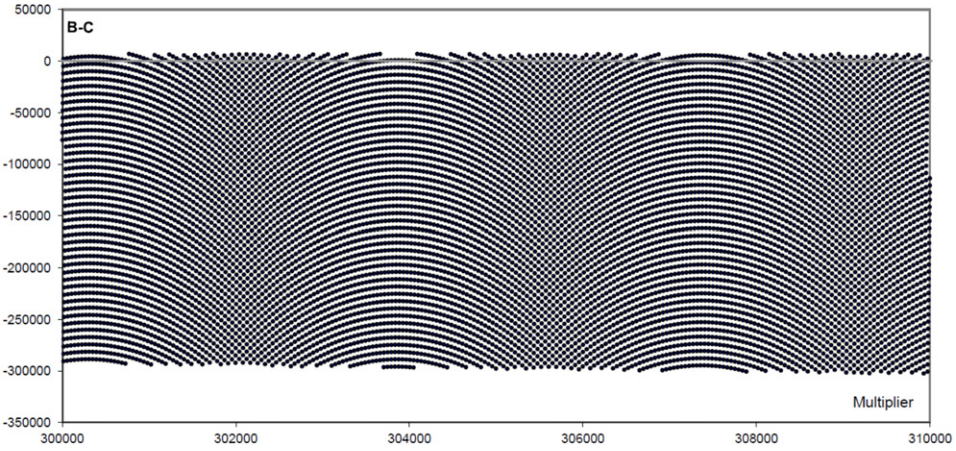

Fig. 3. The distribution of $B - C$ for multipliers from $A = 300000$ to $A = 310000$ in modulus $M = 2^{31} - 1$.

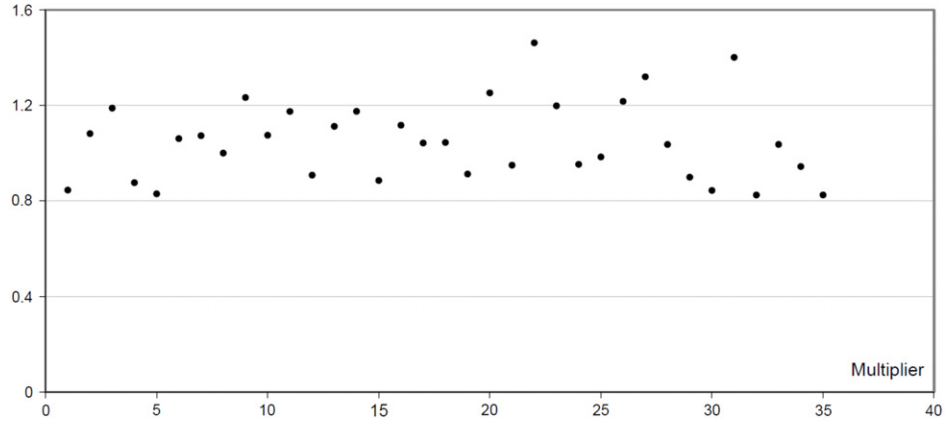

Fig. 4. Distribution of $A/\sqrt{M}$ values for the 35 best multipliers found by Tang and Chang for 64-bit random number generators.

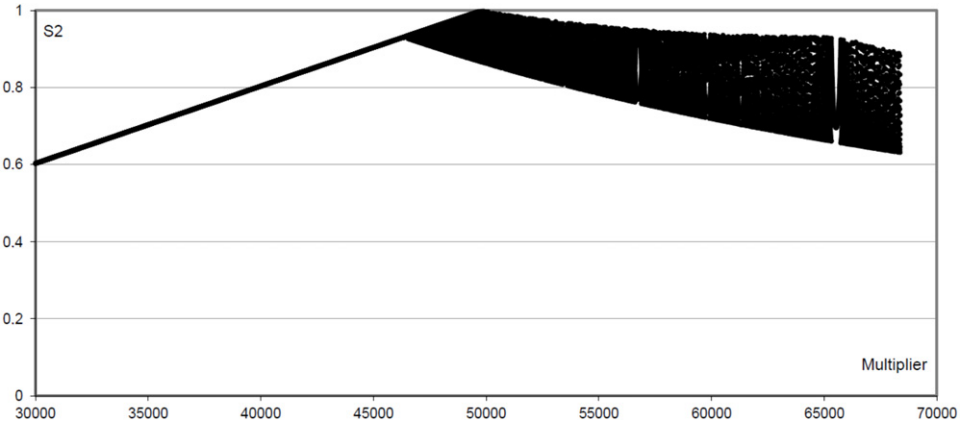

Fig. 5. Distribution of S_2 values for portable multipliers $30000 < A < 68000$ in a generator with $M = 2^{31} - 1$.

S_3 reaches values larger than 0.9. In higher dimensions, the range of bad lattices is negligible. As a result, we suggest searching around the square root of the modulus, where the spectral test value of the two-dimensional space reaches its maximum value and provides the most fertile search area.

5. An application to find 64-bit congruential generators

L'Ecuyer and Simard [15] point out that linear congruential generators with moduli up to 2^{61} fail several tests in the stringent test package TestU01. It would be interesting to investigate the quality of larger moduli by this package. Since 64-bit registers can allow the representation of integer arithmetic up to $2^{63} - 1$, we searched the best portable random number generators for the prime modulus $2^{63} - 25$. All multipliers determined in this study produce full period. If the multiplier A is a primitive root of the modulus M, the generator will have full period when M is prime. This is a necessary and sufficient condition. In our case the factorization of $M - 1$ is $2^{63} - 26 = 2 \times 3^4 \times 17 \times 23 \times 319279 \times 456065899$. Therefore, A is a full-period multiplier in cases where, for all prime factors q, $A^{M-1/q} \neq 1 \text{ Mod}(M)$. In this search, we determined portable full-period multipliers in the top 1% region of a two-dimensional space and investigated their

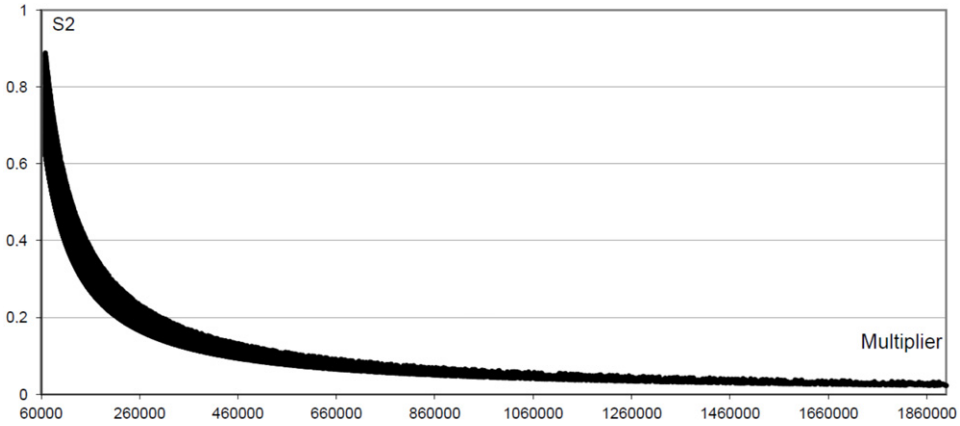

Fig. 6. Distribution of S_2 values for portable multipliers $68000 < A < 2000000$ in a generator with $M = 2^{31} - 1$.

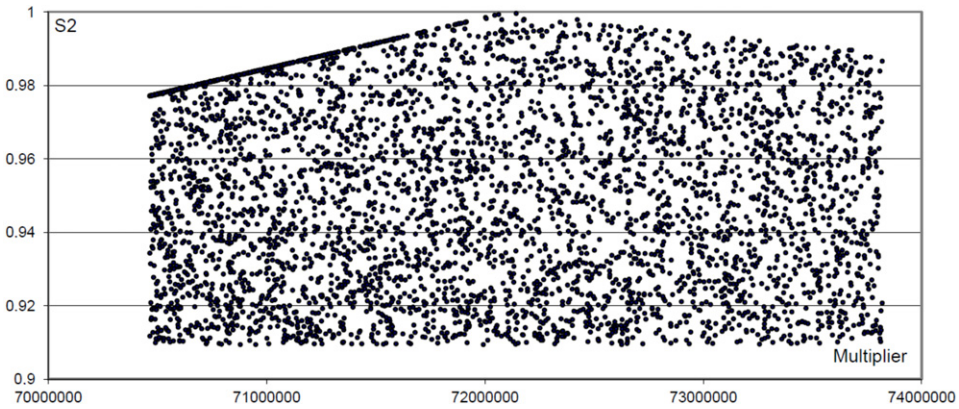

Fig. 7. The spectral values of multipliers having $S_2 > 0.91$ for modulus $M = 4503599090499601$ in a narrow interval around $1.076\sqrt{M}$.

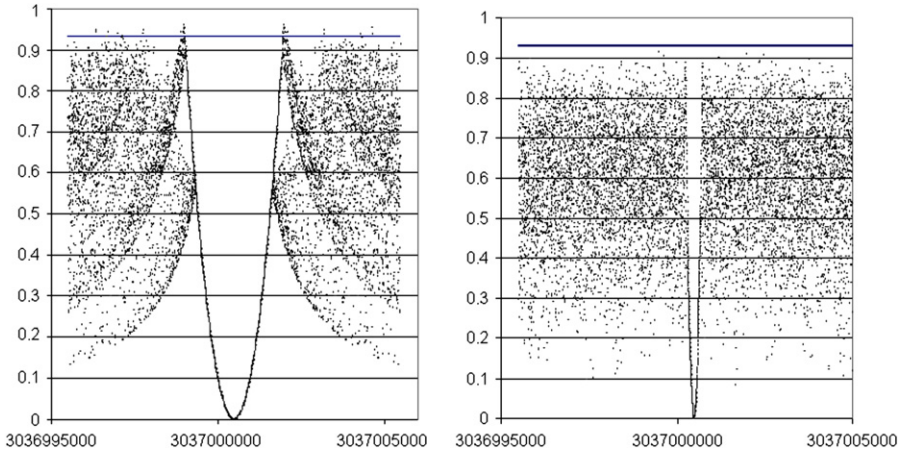

Fig. 8. The comparative distribution of S_2, S_3 (left) and S_2, S_4 values (right) near $\sqrt{M}$ for the 63-bit modulus $M = 9223372012704246017$. In each figure, S_2 is represented by the horizontal line 0.93.

performance in higher dimensions using the respective percentiles of these dimensions. Using prime moduli just less than 2^{63} the top 1% region requires testing multipliers from $A = 3146332518$ to $A = 3525957581$. There are only about 380 million numbers in this interval. The interval of the best 5% multipliers is larger and contains about 1.24 billion numbers. Compared to the size of the modulus, these numbers are still rather modest.

Although CPUs with 64-bit architectures have been around for a long time, software has not caught up with this revolutionary advancement and most of the users are still employing 32 bit compiling facilities. Since the modulus values are very large it is impossible to conduct exhaustive search for 64-bit multipliers. Therefore current literature contains only some partial searches. For example, Dyadkin and Hamilton [16] studied multipliers in the form of $5^n \mod(M)$ for $M = 2^{64}$ under certain restrictions without considering portability. They identified 200 good multipliers and left the investigation of the large prime moduli to be the subject of another study. L'Ecuyer et al. [17] proposed the portable multiplier $A = 2307085864$ for modulus $2^{63} - 25$. But, when subjected to TestU01, this multiplier fails ($p < 10^{-300}$) the Birthday Spacing test in Crush and BigCrush test batteries, and gives a very small p-value ($p = 4.1 * 10^{-8}$) in the Gap $r = 0$ test of BigCrush. L'Ecuyer [18] found six multipliers for the modulus $M = 2^{63} - 25$ without considering portability. Using a fixed worst-case

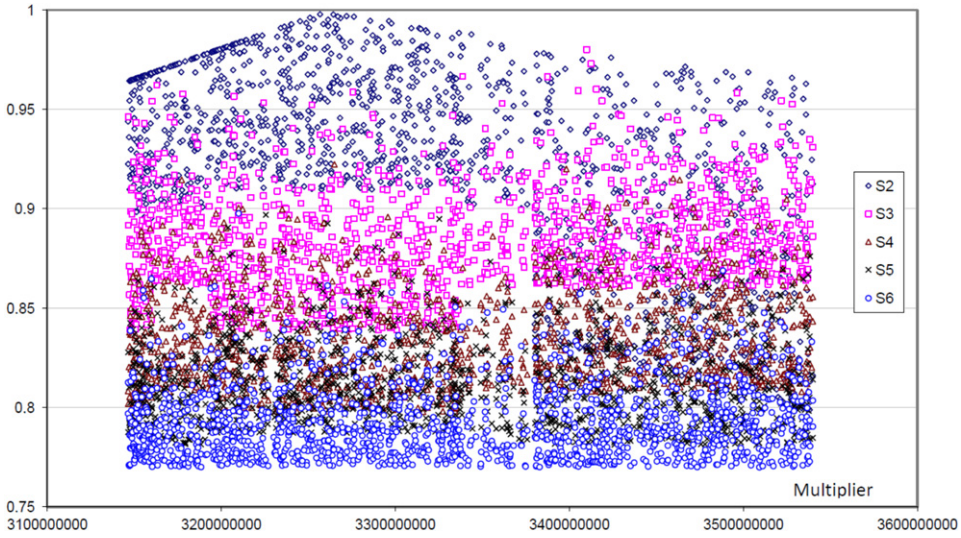

Fig. 9. The distribution of the best 5% spectral test figures of merits S_t for portable multipliers of the prime modulus $M = 2^{63} - 25$ from $A = 3146410910$ to $A = 3539938125$ in dimensions $t = 2, \ldots, 6$.

Table 3
Some useful percentiles of normalized spectral test value S.

Dimension	99%	95%	90%	80%	75%	70%	60%	50%
2	0.964	0.928	0.903	0.851	0.825	0.796	0.738	0.673
3	0.909	0.861	0.824	0.767	0.741	0.716	0.670	0.620
4	0.856	0.808	0.774	0.726	0.705	0.685	0.649	0.610
5	0.825	0.781	0.752	0.714	0.699	0.683	0.654	0.623
6	0.809	0.770	0.744	0.709	0.695	0.680	0.655	0.627
7	0.790	0.751	0.728	0.696	0.686	0.674	0.649	0.629
8	0.768	0.734	0.714	0.686	0.678	0.667	0.643	0.626

performance threshold $S_T = \min_{1 \le t \le T} d^*_t / d_t$, Tang and Chang [13] conducted an exhaustive search for portable generators for three different prime modulus values implemented in 64-bit register size. Since all portable candidate multipliers reported by them failed the TestU01, they resorted to double-precision floating point multipliers and determined two recommendable multipliers.

To the best of our knowledge, there are only few good congruential random number generators suitable for 64-bit computers and compilers. Moreover none of them are portable. Although there are many popular portable generators for 32-bit word-size, there is an increasing demand for 64-bits. Hence we took the prime modulus $M = 2^{63} - 25$ and conducted a systematic search as described in Section 3, evaluating approximately 394 million multipliers between 3 146 410 910 and 3 539 938 125. Using the spectral test we examined dimensions from 2 to 6 and eliminated multipliers falling below the 90th percentiles of the corresponding dimensions. Hence, we were able to choose the best 10% multipliers for each dimension. Rather than using the fixed worst-case performance threshold S_T, we considered percentiles for each dimension independently. As proved by Sezgin [12], S_T places greater emphasis on higher dimensions and neglects the most important part, the smaller dimensions. Instead of a single constant, we used the percentiles presented in Table 3 for assessing the performance of the multipliers. According to the table, a fixed threshold about $S_T = 0.768$ will choose the best 1% multipliers in dimension 8, but will cause a deterioration in dimension 2, leading to the choice of multipliers below the upper 35% group. Almost all available multiplicative congruential generators fail extensive randomness tests in TestU01 irrespective of the modulus size. This may be due to the use of fixed thresholds in spectral test implementations.

By using percentile thresholds listed in Table 3, we obtained 1087 portable generators, all above the 90th percentile. There were 504 generators all above the 95th percentiles of dimensions $t = 2, \ldots, 6$. Fig. 9 depicts spectral test results S_t for these best multipliers. The scatter of S_2 values exhibit the same remarkable pattern observed in Figs. 5 and 7. An interesting property of the S_t values is the formation of distinct layers by the swarm of points in different dimensions. This pattern explains the unsoundness of employing a single fixed threshold for all t values.

Of these 504 multipliers, we filtered 67 multipliers with very large spectral values in several dimensions. They were chosen based on the following criteria:

1. Falling above 97.5th percentiles in all dimensions,
2. or being within the best 1% group in three or more dimensions and not below the 95th percentiles in the remaining dimensions.

We subjected 32 of these 67 multipliers to the Crush and BigCrush test batteries of TestU01, and summarized the results as p-values in Table 4 according to the criteria of L'Ecuyer and Simard [15]. In the table A, B, and C values are also presented for the equation $M = AB + C$.

The results indicate that:

1. Twelve multipliers are successful in all tests of Crush and BigCrush packages. These can be recommended for sophisticated and very demanding applications. In the Result column we labeled them as "Passed all". These good generators can also be used for combined generators with components from different families that are very successful in the above-mentioned test package.
2. The multipliers numbered 5, 10, 14, 17, 18, 20, 21, 27, 29, 30, and 31 are within $(10^{-4}, 1 - 10^{-4})$ interval and can be considered "Pass" according to the TestU01 criteria of L'Ecuyer and Simard [15]. In the Result column, these are labeled as "Passed".
3. Although not definitively rejected, the remaining nine multipliers can be considered as "suspect" since they have p-values within the intervals $(10^{-10}, 10^{-4}]$ and $[1 - 10^{-4}, 1 - 10^{-10})$. These are multipliers 1, 2, 3, 6, 9, 13, 22, 23, and 24.

We must emphasize that the listed multipliers are only a small portion of good generators and by conducting TestU01 to the remaining multipliers, and using different filtering criteria it will be possible to obtain several good generators.

Although there are some 64-bit random number generators available, multiplicative congruential generators remain still popular due to their well-documented theory, ease of implementation and widespread applications in program packages. In

Table 4
Results of Crush and BigCrush test batteries applied to 32 multipliers for the modulus $M = 2^{63} - 25$. The values under the Crush and BigCrush columns indicate the p-values for failed test instances.

No	A	B	C	Crush	BigCrush	Result
1	3154053667	2924291407	1229836314		0.9993, 0.9999	Suspect
2	3157107955	2921462353	1965457668		1−4.1e−6	Suspect
3	3159143104	2919580320	2352662503		6.7e−5, 7.8e−4, 2.7e−4, 0.9995	Suspect
4	3163036175	2915986895	2143849158			Passed all
5	3163786827	2915295037	1975698184		1.8e−4, 5.0e−4	Passed
6	3172190117	2907572275	1636569608		4.4e−5, 1.5e−5	Suspect
7	3200261722	2882068042	1842687459			Passed all
8	3201541663	2880915823	1924342134			Passed all
9	3206549749	2876416322	1526172605		6.7e−5, 0.9995	Suspect
10	3206832497	2876162707	1387686404		0.9993, 7.2e−4	Passed
11	3211103532	2872337171	1961787811			Passed all
12	3213258092	2870411206	1807796831			Passed all
13	3217568780	2866565617	1774138523	1.7e−5, 9.7e−4, 7.2e−4		Suspect
14	3238858873	2847722731	1711633620		3.4e−4	Passed
15	3245854730	2841584976	1808239303			Passed all
16	3261037634	2828354981	1502420829			Passed all
17	3273091456	2817938991	1883414887	8.3e−4, 0.9994		Passed
18	3277628277	2814038462	1237985809		0.9993	Passed
19	3286706186	2806266065	1457397693			Passed all
20	3312958483	2784028862	1575039437		2.6e−4	Passed
21	3338736601	2762533598	1719955385		0.9994	Passed
22	3352494981	2751196374	1274376889	9.2e−5	1.6e−4	Suspect
23	3363261634	2742389097	1414771285		2.7e−5	Suspect
24	3393139931	2718240987	1784223886		1−5.8e−5	Suspect
25	3423977237	2693759741	1723760166			Passed all
26	3459480860	2666114486	1969037823			Passed all
27	3464484710	2662263744	1779421543	0.9995	0.9992, 7.1e−4, 3.1e−4	Passed
28	3465965455	2661126360	1704881983			Passed all
29	3474009732	2654964363	1679595067		0.9996, 6.6e−4, 5.1e−4	Passed
30	3474801229	2654359610	1818815093	3.7e−4		Passed
31	3512389242	2625953845	1688240293		9.7e−4	Passed
32	3512424704	2625927333	1516741351			Passed all

Listing 2, we present the C code of our generator with multiplier $A = 3163036175$.

```
    double sezgin64 (void)
{
    IX = 3163036175LL*( IX % 2915986895LL )
    - 2143849158LL*(IX/2915986895LL);
    if ( IX <0LL ) { IX = IX + 9223372036854775783LL; }
    return IX*1.084202172485504437e-19;
}
```

Listing 2. The C code of 64-bit multiplicative congruential random number generator.

Speed is an important criterion in large scale Monte Carlo studies. We compared the speed of our generator with the following well-known generators applicable in 64-bit arithmetic:

1. Combined Multiple Recursive Generator MRG63k3a proposed by L'Ecuyer [19],
2. The enhanced version of Wichmann and Hill Generator [20],
3. 64-bit Linear Feedback Shift Register Generator lfsr258 by L'Ecuyer [21],
4. Mersenne Twister MT19937 proposed by Matsumoto and Nishimura [22],
5. ISAAC64 developed for cryptography by Jenkins [23],
6. 64-bit Universal RNG by Marsaglia and Tsang [24].

Generators were implemented in C, compiled with gcc version 4.5.3, and run on an Intel Core i7 3.07 GHz CPU with 8.00 GB Ram under the 64-bit Professional Edition of the Windows 7 operating system. The time taken by 10^8 calls to the generator functions are presented in Table 5. Our generator is faster than most of the present popular generators, including the Mersenne Twister. Table 5 shows that, our generator has excellent runtime characteristics in terms of speed, and since it behaves very well in stringent empirical statistical tests, it is recommendable for general use.

Table 5
CPU time (seconds) for 10^8 calls of the random number generators.

Generator	Time (s)
MRG63k3a	13.462
Wichmann and Hill	10.435
lfsr258	4.695
Mersenne Twister MT19937	3.650
Sezgin64	3.100
ISAAC64	2.183
64-bit Universal	2.058

Moreover, the LFSR258 and Mersenne Twister generators respectively fail 6 and 2 of the tests in the Crush and BigCrush test batteries of TestU01, whereas our generators pass all tests.

6. Results and future work

Finding high-quality multipliers is an essential part of developing multiplicative random number generators. The first step should be finding multipliers suitable for approximate factoring. For a modulus M and multiplier A, we define $B = \lfloor M/A \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Letting $C = M - AB$, approximate factoring can be used for obtaining portable generators if $B - C > 0$. We observed that all multipliers less than the square root of the modulus are portable. Above $\sqrt{M}$, most of the time we may have $B - C < 0$. In this case, A must be replaced by $\lfloor M/B \rfloor$. The last portable multiplier is $\lfloor M/2 \rfloor$.

After determining multipliers portable by approximate factoring, the second step is checking for the full-period condition. In the third step, the figure of merits obtained in the spectral test must be evaluated. The best spectral test results are obtained near $\sqrt{M}$ for a two-dimensional space. Although there is a bad lattice region for higher dimensional spaces near $\sqrt{M}$, the length of the interval

is negligibly short compared to the magnitude of the modulus. In this paper, we supplied some of the best percentile ranges for good multipliers.

As pointed out by L'Ecuyer and Simard [15] linear congruential generators with moduli up to 2^{61} fail several tests in the stringent test package TestU01. As a future objective, the best portable random number generators could be investigated for other moduli of sizes 2^{64}, 2^{128} or larger. In these searches, portable full-period multipliers in the top 1% of a two-dimensional space may be determined, and their performance investigated in higher dimensions using the respective percentiles of these dimensions. We already presented 12 excellent portable generators suitable for 64-bit implementation. They passed all tests of TestU01 Crush and BigCrush tests successfully and can be used for demanding applications. We also plan to present a parallel application.

There is also a need to investigate congruential generators for moduli larger than 2^{64} (e.g., moduli in the order of 2^{128}). For example $M = 2^{127} - 1$ is Mersenne prime and the best multipliers can be searched near $A = 1.076 * 2^{63.5} = 1.4035 * 10^{19}$.

As another future extension of this research, one can work on the application of the search method for good multipliers for combined and large order Multiple Recursive Generators.

Acknowledgments

We would like to thank the Principal Editor David P. Landau and the anonymous referee for their prompt processing and constructive comments that improved the earlier version of this paper.

References

[1] J.E. Gentle, Computer implementation of random number generators, Commun. ACM 31 (1990) 119–125.
[2] C.D. Kemp, The construction of fast portable multiplicative congruential random number generators, Commun. ACM 39 (1996) es. Electronic Supplement to the December issue. Article no. 163.
[3] P. L'Ecuyer, S. Cote, Implementing a random number package with splitting facilities, ACM Trans. Math. Softw. 17 (1991) 98–111.
[4] K. Marse, S.D. Roberts, Implementing a portable FORTRAN uniform (1, 1) generator, Simulation 41 (1983) 135–139.
[5] W.H. Payne, J.R. Rabung, T.P. Bogyo, Coding the Lehmer pseudo-random number generator, Commun. ACM 12 (1969) 85–86.
[6] L. Schrage, A more portable Fortran random number generator, ACM Trans. Math. Softw. 5 (1979) 132–138.
[7] F. Sezgin, A method of obtaining portable random number generators, in: Proc. COMPSTAT'88. Short Communications and Posters. Copenhagen, Denmark, 1988, pp. 41–42.
[8] F. Sezgin, Some comments on computer implementation of random number generators, J. Comput. Appl. Math. 39 (1992) 383–386.
[9] B.A. Wichmann, I.D. Hill, An efficient and portable pseudorandom number generator, Appl. Stat. 31 (1982) 188–190.
[10] S.K. Park, K.W. Miller, Random number generators: good ones are hard to find, Commun. ACM 31 (1988) 1192–1201.
[11] D.E. Knuth, The Art of Computer Programming, Vol. 2, third ed., Addison-Wesley, Reading, MA, 1998.
[12] F. Sezgin, Distribution of lattice points, Computing 78 (2006) 173–193.
[13] H.C. Tang, H. Chang, An exhaustive search for good 64-bit linear congruential random number generators with restricted multiplier, Comput. Phys. Comm. 182 (2011) 2326–2330.
[14] F. Sezgin, A method of systematic search for optimal multipliers in congruential random number generators, BIT 44 (2004) 135–149.
[15] P. L'Ecuyer, R. Simard, TestU01: AC library for empirical testing of random number generators, ACM Trans. Math. Softw. 33 (2007). Article 22.
[16] I.G. Dyadkin, K.G. Hamilton, A study of 64-bit multipliers for Lehmer pseudorandom number generators, Comput. Phys. Comm. 103 (1997) 103–130.
[17] P. L'Ecuyer, F. Blouin, R. Couture, A search for good multiple recursive random number generators, ACM Trans. Model. Comput. Simul. 3 (1993) 87–98.
[18] P. L'Ecuyer, Tables of linear congruential generators of different sizes and good lattice structure, Math. Comp. 68 (1999) 249–260.
[19] P. L'Ecuyer, Good parameters and implementations for combined multiple recursive random number generators, Oper. Res. 47 (1999) 159–164.
[20] B.A. Wichmann, I.D. Hill, Generating good pseudo-random numbers, Comput. Statist. Data Anal. 51 (2006) 1614–1622.
[21] P. L'Ecuyer, Tables of maximally equidistributed combined LFSR generators, Math. Comp. 68 (1999) 261–269.
[22] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Trans. Model. Comput. Simul. 8 (1998) 3–30.
[23] B. Jenkins, ISAAC, in fast software encryption, in: D. Gollmann (Ed.), Proceedings of the 3rd International Workshop, Cambridge, UK, in: Lect. Notes Comput. Sc., vol. 1039, Springer-Verlag, 1996, pp. 41–49.
[24] G. Marsaglia, W.W. Tsang, The 64-bit universal RNG, Statist. Probab. Lett. 66 (2004) 183–187.