View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Journal of Computational and Applied Mathematics 236 (2012) 3228–3237

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

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

Efficient computer search of large-order multiple recursive pseudo-random number generators

Lih-Yuan Deng[a], Jyh-Jen Horng Shiau[b,*], Henry Horng-Shing Lu[b]

[a] Department of Mathematical Sciences, The University of Memphis, Memphis, TN 38152, USA

[b] Institute of Statistics, National Chiao Tung University, Hsinchu, Taiwan, 30010, ROC

ARTICLE INFO

Article history:
Received 26 October 2009
Received in revised form 9 February 2012

Keywords:
Empirical tests
Equi-distribution
Factorization
Portable and efficient generators
DX/DL/DS generators
Primality testing

ABSTRACT

Utilizing some results in number theory, we propose an efficient method to speed up the computer search of large-order maximum-period Multiple Recursive Generators (MRGs). We conduct the computer search and identify many efficient and portable MRGs of order up to 25,013, which have the equi-distribution property in up to 25,013 dimensions and the period lengths up to $10^{233,361}$ approximately. In addition, a theoretical test is adopted to further evaluate and compare these generators. An extensive empirical study shows that these generators behave well when tested with the stringent Crush battery of the test package TestU01.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Maximum-period Multiple Recursive Generators (MRGs) of order k have become popular pseudo-random number generators (PRNGs) in many areas of applications because of the great properties of equi-distribution over spaces up to k dimensions, long periods, and excellent empirical performances. As the order k gets larger, these nice properties get more profound; but, unfortunately, finding maximum-period MRGs also gets more difficult. In this study, we utilize some number theory results to accelerate the computer search and find a number of large-order maximum-period MRGs accordingly.

An MRG of order k is defined by a k-th order linear recurrence with modulus p that corresponds to a k-th degree polynomial. This MRG has the maximum period if and only if the corresponding polynomial is a primitive polynomial. For maximum-period MRGs, we discuss some issues related to the large order in Section 2. In Section 3, we discuss search algorithms for large-order maximum-period MRGs, in which the key issue is to develop a method that can find a k-th degree primitive polynomial efficiently. Alanen and Knuth [1] and Knuth [2] gave three necessary and sufficient conditions for primitive polynomials. Implementation of direct checking of these classical conditions would incur two bottlenecks in computation. The first bottleneck incurs because no early exit mechanism is provided for non-primitive (or reducible) polynomials when checking the conditions; thus a tremendous amount of computing time would be wasted. The other bottleneck resides in the difficulty of factoring a large integer, $(p^k - 1)/(p - 1)$, which fortunately can be bypassed by solving an "easier" problem of testing its primality; see [3]. Bypassing these bottlenecks, Deng [4] developed an efficient search algorithm (referred to as Algorithm GMP hereafter) with a built-in early exit strategy.

However, if the degree k gets too large, primality testing of $(p^k - 1)/(p - 1)$ itself would become a major bottleneck as well. Therefore, in this paper, we adopt an alternative approach—quickly rule out, for each prime order k considered, the prime modulus p for which $(p^k - 1)/(p - 1)$ has "small" (say less than 10^{12}) prime factors. Some classical number theory

* Corresponding author. Tel.: +886 3 5731884; fax: +886 3 5728745.
E-mail addresses: jyhjen@stat.nctu.edu.tw, jyhjen@gmail.com (J.-J.H. Shiau).

0377-0427/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2012.02.023

results, in particular Legendre's Theorem, are utilized to speed up the computer search. With the successfully found prime modulus p and the associated order k such that $(p^k - 1)/(p - 1)$ is prime, we can then search for large-order MRGs using Algorithm GMP of [4].

Not all MRGs are efficient in generating random numbers or having good theoretical/empirical performances. Using Algorithm GMP to search for good maximum-period MRGs, it is necessary to restrict the search within some special classes of MRGs. In Section 4, we describe and discuss three classes of efficient large-order MRGs called DX/DL/DS generators, which were proposed and/or discussed in [5–7]. Having conducted an extensive computer search using the accelerated method proposed in this paper, we obtain and list a number of new efficient and portable DX/DL/DS MRGs of several large orders. The largest order k of the MRGs found so far is $k = 25{,}013$ with the period length approximately $10^{233,361}$. We further conduct an extensive empirical study on the listed large-order MRGs and find that all of them pass the stringent tests in the Crush battery of the TestU01 test suite of [8]. Although most applications at the present time may not need PRNGs with period lengths as long as these MRGs, these MRGs may come in handy for very-large-scale simulation applications in the future. In addition, to further compare the new generators with the same equi-distribution dimensions, say, up to the order k, we adopt a theoretical test to evaluate their lattice structure over $k + 1$ dimensions.

Throughout this paper, we let p be a prime number and $\mathbb{Z}_p \equiv \{0, 1, \ldots, p - 1\}$ be the finite field of p elements under the usual operations of addition and multiplication with modulus p.

2. Large-order multiple recursive generators

An MRG generates pseudo random numbers sequentially with a k-th order linear recurrence:

$$X_i = (\alpha_1 X_{i-1} + \cdots + \alpha_k X_{i-k}) \bmod p, \quad i \geq k, \tag{1}$$

where the not-all-zero multipliers $\alpha_1, \ldots, \alpha_k$ and not-all-zero initial seeds $X_0, \ldots, X_{k-1}$ are integers in $\mathbb{Z}_p$, and the prime modulus p is a positive integer. It is well known that the maximum period of the MRG in (1) is $p^k - 1$, which is achieved if and only if its characteristic polynomial

$$f(x) = x^k - \alpha_1 x^{k-1} - \cdots - \alpha_k \tag{2}$$

is a primitive polynomial. When $k = 1$, the MRG reduces to the linear congruential generator (LCG) proposed in [9].

2.1. Advantages of large-order MRGs

Choosing a large order k for a maximum-period MRG has several important advantages:

1. It has an extremely long period length, $p^k - 1$, which goes up exponentially in k. Knuth [2, page 95] commented that a generator with longer period should be used even though the application only needs a small fraction of the period, because it would have better "accuracy" in higher dimensions, which leads to a more effective number of bits of precision.
2. Knuth [2, page 30] also commented that "*all known evidence indicates that the result will be a very satisfactory source of random numbers*" for the sequence generated by an MRG (with only a small order then). In the development of large-order MRGs, various extensive empirical evaluation studies had already shown that large-order MRGs tend to have better empirical performance than small-order MRGs; see, for example, [8,10]. With no doubt, the large-order MRGs found in this paper also have excellent empirical performances (shown in Section 4).
3. A maximum-period MRG is equi-distributed up to k dimensions as stated in [11, Theorem 7.43]: every t-tuple ($1 \leq t \leq k$) of integers between 0 and $p - 1$ appears exactly the same number of times (i.e., p^{k-t}) over its entire period $p^k - 1$, with the exception of the all-zero tuple that appears one time less (i.e., $p^{k-t} - 1$).

We remark that it does not take a very large order k to have a long enough period even for current very-large-scale applications; many published MRGs can serve the purpose. Nonetheless, the advantage of achieving higher equi-distribution dimensions so as to have better PRNGs in terms of the lattice structure makes searching for MRGs of higher order worth pursuing. The extremely long period is merely a side-product.

2.2. Theoretical test to compare large-order MRGs

Consider a t-tuple $(X_i, X_{i+1}, \ldots, X_{i+t-1})$ with the index i running through the whole period $(p^k - 1)$ of a maximum-period MRG in (1). For any $t \leq k$, by the equi-distribution property, we can see that a large-order maximum-period MRG is pretty close to an "ideal" generator that produces all p^t t-tuples with equal frequency for any value of t. When $t > k$, the equi-distribution property is impossible to achieve because the number of possible t-tuples, p^t, is larger than the period length of the MRG. In fact, like the well-known problem for the LCG [12], these t-dimensional points (vectors) lie on a relatively small family of equidistant parallel hyperplanes in a high dimensional space; see, for example, [13,2]. Let $d_t(k)$ denote the the maximum distance between two adjacent hyperplanes, taken over all families of parallel hyperplances that cover all such t-tuples. If $d_t(k)$ is large, then the generator is considered "bad" because the "gap" between two adjacent hyperplanes in t-dimensional space is wide. Clearly, if the dimension t is much larger than k, the maximum gap $d_t(k)$ becomes so large that no MRG (of fixed order k) can be considered "good". Performance evaluation of a given generator based on $d_t(k)$'s is often called the spectral test in the literature; see, for example, [2].

The popular generator MT19937, proposed in [14], has a property of 623-dimension equi-distribution. In contrast, the generators reported later in this paper have much higher dimensions of equi-distribution ranging from 11,003 to 25,013.

Because all maximum-period MRGs have the "perfect" lattice structure for dimensions up to the order k, the differentials on the lattice structure among the generators of the same order k can only reside in dimensions $t \geq k + 1$. If using $d_t(k)$ as the figure of merit to compare generators, one would want this value to be as small as possible. A well-known lower bound of $d_t(k)$ is

$$d_t^*(k) = \begin{cases} \dfrac{p^{-k/t}}{r_t}, & t > k, \\ \dfrac{1}{p}, & t \leq k, \end{cases}$$

where the constant r_t depends only on t; see, for example, [2,15,16]. As pointed by a referee, the bound given above "is one that can be reached exactly for a general lattice in the real space; in general it cannot be reached for a fixed p and k". If the exact value of $d_t^*(k)$ is known, Fishman and Moore [17] suggested to compare generators with different values of modulus by normalizing $d_t(k)$ with $d_t^*(k)/d_t(k)$ so that the value is between 0 and 1, the larger the better. Then a good value for a generator is one that is not too far away from 1. Unfortunately, the value of r_t is known only for $t \leq 8$; and so far as we know, when t is large, there is no general formula for r_t.

At the request of a referee, we perform a spectral test to compare among the MRGs found in this paper with the same order k (and the same modulus p). The reason for only comparing the MRGs with the same order is that, when all other conditions stay the same, a maximum-period MRG of a smaller order k_1 is expected to be inferior to a maximum-period MRG of a larger order k_2 in the lattice test for dimension t whenever $k_1 < t \leq k_2$; and this is simply because the MRG of the larger order k_2 has the equi-distribution property for dimension t, whereas the MRG of the smaller order k_1 has not.

To compare performances among generators, one can consider various figures of merit for the spectral test but results may not always be consistent. In this study, we adopt a simple figure of merit $d_{k+1}(k)$, the maximum gap between adjacent hyperplanes in the $(k + 1)$-dimensional space, to compare MRGs of order k. The key for us to be able to compute $d_{k+1}(k)$ for k as large as 25,013 is that the $d_{k+1}(k)$ value of an MRG with c nonzero terms can be obtained by solving a $(c + 1)$-variate quadratic integer program. For example, see [16] for an explicit formulation. We should note that a similar idea for computing/approximating the spectral values of MRGs with very few nonzero terms was also considered by L'Ecuyer and his collaborators; see, for example, [13,18].

We remark that because the dimensions of equi-distribution of the reported MRGs are all so large, the actual impact or practical effect of various $d_t(k)$ values on the actual performance of the generators would be very minimal, and our empirical study as reported in Section 4.3 confirms that.

2.3. Potential problems with large-order MRGs

On the negative side, an MRG with a large order k requires a memory space of size k to store the k-states of the MRG. However, the memory cost is drastically reduced nowadays and is considered minimal with respect to the current capacity of computers, even with k as large as 30,000. Also, the one-time cost for the initialization of k states is minimal, especially for a large-scale simulation study. On the other hand, increasing the order k by 1 will increase the period length by p folds. Nonetheless, if this memory space requirement or initialization cost is an issue for the application, one can certainly consider a smaller k.

To construct a maximum-period MRG of order k, we need a k-th degree primitive polynomial. As mentioned earlier, the task of searching for k-th degree primitive polynomials gets more and more difficult as the order k gets larger. Therefore, an efficient search algorithm plays an important role in finding primitive polynomials successfully when the order k is large. We describe some search algorithms next.

3. Search algorithms for primitive polynomials

As mentioned earlier, Alanen and Knuth [1] and Knuth [2] gave three conditions that are necessary and sufficient for a k-th degree characteristic polynomial $f(x)$ in (2) to be a primitive polynomial. However, when the order k or the prime modulus p is large, some of the conditions can be very difficult to verify. For example, one condition requires complete factorization of $(p^k - 1)/(p - 1)$; and it is still extremely hard to factor a general integer of 200 (or more) digits even with modern technologies. As pointed out in [2, page 30], factorization of $(p^k - 1)/(p - 1)$ is indeed the limiting factor in calculation to carry out the test for primitivity modulo p. To avert the difficulty of factoring

$$R(k, p) \equiv (p^k - 1)/(p - 1), \tag{3}$$

one can search for p with a fixed k such that $R(k, p)$ is a prime number. Clearly, k has to be an odd prime number.

There are some studies in the field of number theory regarding the primes of the form $(a^k - 1)/(a - 1)$, in particular for small values of a (from 2 up to 12). For example, when $a = 2$, $R(k, 2)$ is a Mersenne number; when $a = 10$, it is a decimal integer with all 1's, called a repunit number in the literature; see, for example, [19,20].

Finding a large prime p with primality of $R(k, p)$ in (3) as a means to construct random number generators (RNGs) was first considered in [3] for $k \leq 7$ and later in [21] for $k \leq 13$. Deng [4] found some prime numbers of the form $R(k, p)$ in (3) for large k up to 1511 and named the class as the *Generalized Mersenne Primes (GMPs)*. Recently, Deng [22] listed more GMPs with even larger k's (up to 10,007).

With the found prime modulus p for GMPs, Deng [4] proposed an efficient search algorithm called *Algorithm GMP*, which will exit earlier when $f(x)$ in (2) is not a primitive polynomial. Our experience indicates that the early strategy saves tremendous amount of time especially when k is large.

With Algorithm GMP, to search for a maximum-period MRG with a large prime order k efficiently, it remains to have a method that can quickly find a prime modulus p such that $(p^k - 1)/(p - 1)$ is also a prime. Next, we develop such a method.

3.1. Theory about prime factors of $(p^k - 1)/(p - 1)$

As k increases, the likelihood of finding a p such that $R(k, p)$ is a prime by computer search decreases and the computing time for some probabilistic primality testing procedures increases drastically. According to our evaluation, the running time for testing one $R(k, p)$ with $k = 10{,}007, 15{,}013, 20{,}011$, and $25{,}013$ are 701.5, 1768.4, 3194.7, and 5380.5 seconds, respectively, on some personal computers with 3.2 GHz Intel Xeon processors and 2MB L2 cache. The problem with most of the probabilistic primality testing programs is that, regardless whether the number under test is a prime or not, the amount of computing time is about the same. An early exit strategy to quickly screen out some p's with a composite number $R(k, p)$ would help speed up the search. For this, with the help of some classical number theory results, we develop an efficient screening procedure as described below.

The following theorem gives a characterization of the prime factors of $p^k - 1$ (or $(p^k - 1)/(p - 1)$). It is a special case of Legendre's Theorem for the prime factors of the form $a^k \pm b^k$, where a, b are integers with $\gcd(a, b) = 1$ and k is any positive integer.

Theorem 1 (*Theorem 2.4.3 in [23, page 41]*)**.** *Let k be a prime. For every prime factor q of $(p^k - 1)/(p - 1)$, $q = 1 \bmod k$.*

From Theorem 1, it is easy to see that any prime factor of $(p^k - 1)/(p - 1)$ must be of the form $2kc + 1$, because both q and k are odd primes. Therefore, we can rule out any prime number such that $q \neq 1 \bmod 2k$ to be a factor of $(p^k - 1)/(p - 1)$. This simple fact had been utilized in [24] in obtaining the complete factorization of $p^k - 1$ for $k = 7499$ and $k = 20{,}897$ with $p = 2^{31}-1$. With the obtained complete factorizations, the authors were able to find several efficient and portable MRGs of order $k = 7499$ and 20,897 through computer search. In the current paper, we are looking for prime modulus p such that $(p^k - 1)/(p - 1)$ is a prime. With the help of Theorem 1, we can greatly speed up the computer search by skipping all primes $q \neq 1 \bmod 2k$. We can further quantify the savings with the help of another powerful theorem in the number theory described in the following.

Unless stated otherwise, we assume k is an odd prime number. For a given n, denote the set of all primes below n by

$$\mathbb{P}_n \equiv \{q | q \leq n, q \text{ is a prime}\} \tag{4}$$

and the subset of our particular interest by

$$\mathbb{Q}_{k,n} \equiv \{q \in \mathbb{P}_n | q = 1 \bmod k\}. \tag{5}$$

Theorem 1 indicates that, for any prime factor q of $(p^k - 1)/(p - 1)$, $q \in \mathbb{Q}_{k,n}$ for some integer n. Therefore, we can search for prime factors of $R(k, p)$ in the set $\mathbb{Q}_{k,n}$, a set much smaller than $\mathbb{P}_n$. The following theorem gives a general result on the relative size between $\mathbb{Q}_{k,n}$ and $\mathbb{P}_n$. It is a well-known result from a general property of Arithmetic Progressions; see, for example, [25, pages 61–62]. Let $\phi(x)$ be the Euler's phi function, the number of integers between 1 and x that are relatively prime to x. For a simple formula to compute $\phi(x)$, see [2]. Denote the number of elements in a set S by $|S|$.

Theorem 2. *Let $k > 1$ be an integer and $\mathbb{P}_n$ and $\mathbb{Q}_{k,n}$ be defined as in (4) and (5), respectively. We have*

$$\lim_{n\to\infty} \frac{|\mathbb{Q}_{k,n}|}{|\mathbb{P}_n|} = \frac{1}{\phi(k)}.$$

In particular, for a prime k,

$$\lim_{n\to\infty} \frac{|\mathbb{Q}_{k,n}|}{|\mathbb{P}_n|} = \frac{1}{k-1}.$$

The asymptotic behavior of $|\mathbb{P}_n|$ "has been studied extensively by many of the world's greatest mathematicians, beginning with Legendre in 1798" as commented in [2]. In the following, for easy reference, we use the common notation $\pi(n)$ for $|\mathbb{P}_n|$ and quote some useful results stated in [2, pages 379–383] without giving specific citations.

Table 1
List of k, c, and $p = 2^{31} - c$, for which $(p^k - 1)/(p - 1)$ is a prime.

k	c	$p = 2^{31} - c$	$\log_{10}(p^k - 1)$
5 003	1 259,289	2146,224,359	46686.4
6 007	9 984,705	2137,498,943	56044.7
7 001	610,089	2146,873,559	65332.0
8 009	5 156,745	2142,326,903	74731.1
9 001	7 236,249	2140,247,399	83983.5
10,007	431,745	2147,051,903	93383.7
11,003	1 276,425	2146,207,223	102676.4
12,007	37,532,781	2109,950,867	111956.5
13,001	292,495	2147,191,153	121323.7
14,009	626,301	2146,857,347	130729.2
15,013	8 996,265	2138,487,383	140072.9
20,011	26,131,941	2121,351,707	186634.9
25,013	11,538,909	2135,944,739	233361.1

According to [2], $\pi(n)$ can be approximated fairly well by

$$L(n) \equiv \int_2^n \frac{dt}{\ln t},$$

or further by

$$R(n) \equiv L(n) - \frac{1}{2}L(\sqrt{n}) - \frac{1}{3}L(\sqrt[3]{n}),$$

when n is of reasonable size (say, $n \le 10^{18}$). As an illustration, for $n = 10^{12}$,

$$\pi(n) \equiv |\mathbb{P}_n| = 37{,}607{,}912{,}018, \qquad R(n) = 37{,}607{,}910{,}542, \quad \text{and} \quad L(n) = 37{,}607{,}950{,}280.$$

From Theorem 1, we know that any prime factor q satisfies $q = 1 \bmod k$. Since q is an odd number, it can be easily shown that $q = 1 \bmod 2k$ for $k > 2$. Thus, when k is an odd prime, $\mathbb{Q}_{k,n} = \mathbb{Q}_{2k,n}$. By searching factors through $\mathbb{Q}_{2k,n}$ instead of $\mathbb{P}_n$, Theorem 2 tells us that we skip about $(k-2)/(k-1)$ of the q's for which $q \ne 1 \bmod 2k$. For illustration, we find that there are 1503,440 primes in $\mathbb{Q}_{2k,n}$ (of the form $2kc + 1$) for $k = 25013$ with $n = 10^{12}$. Therefore, we skip a fraction

$$(37{,}607{,}912{,}018 - 1503{,}440)/37{,}607{,}912{,}018 = 0.9999600233$$

of all prime $q \le n$, for which none of them can be a prime factor of $R(k, p)$. This fraction is extremely close to the theoretical limiting value, $(k-2)/(k-1) = 0.9999600192$, as n goes to infinity.

Skipping, say, 99.996% of the primes enables us to compute the product of all the primes in $\mathbb{Q}_{2k,n}$ for a much larger value of n, which leads to a higher chance of finding prime factors of $R(k, p)$ if it is a composite number. Once a prime factor is found, the search program can go on to verify the next candidate p. The following screening procedure is developed based on the aforementioned observations.

3.2. *Procedure for finding GMPs*

We summarize the steps of the procedure for finding proper (p, k) pairs such that $(p^k - 1)/(p - 1)$ is a prime as follows:

1. For a given prime order k, compute Q_n, the product of all prime numbers $q \le n$ of the form $2kc + 1$ with, say, $n = 10^{12}$.
2. For a given prime p, check whether there is any common factor between Q_n and $(p^k - 1)/(p - 1)$:
   (i) if the common factor is larger than one (i.e., the early exit condition is satisfied), then we move on to another p;
   (ii) otherwise, apply the probabilistic prime test (see [26]) to test the primality of $(p^k - 1)/(p - 1)$:
      (a) if the primality test fails, move on to another p;
      (b) otherwise, record the prime modulus p as the result for the current k, and then go on to repeat the whole procedure with the next prime order k.

For the actual search, given a range $[a, b]$ for p and an order k, we used the above steps to search for a prime p within a set where $(p - 1)/2$ is also a prime, a strategy adopted in [3]. For 32-bit RNGs, we started from the upper bound $b = 2^{31}$–1 and moved downward until the search was successful or the lower limit a was reached. Deng [22] listed some p's for which $R(k, p)$ is a GMP for k from 5003 to 10,007 in increments around 1000. In this study, with the skipping strategy described earlier, we were able to extend the search and find GMPs for $k = 11{,}003$, 12,007, 13,001, 14,009, 15,013, 20,011, and 25,013. Table 1 lists these new GMPs as well as the GMPs found in [22] for completeness. The number of searches needed for each p listed in Table 1 appeared fairly random and increasing as k increases in general. In total, several months of CPU time were spent in searching for the new p's listed in Table 1.

With the newly found (p, k) pairs of prime modulus p and the associated order k (via finding GMPs), we then employ Algorithm GMP to search for efficient and portable maximum-period MRGs within DX/DL/DS classes in the next section.

As argued in [22], for a specific choice of k and p, even if an unlikely false passing was made by the probabilistic primality tests, its effect on the successful search of a primitive polynomial using Algorithm GMP is negligible. In other words, once $R(k, p)$ is claimed as a probable prime, it is fairly safe to claim that the primitive polynomial found subsequently by Algorithm GMP is indeed a primitive polynomial and hence a maximum-period MRG is found.

4. New large-order DX, DL, and DS generators

Generally speaking, an MRG is efficient as long as it can be implemented with a recursive equation with just a few terms. In this section, we present some efficient and portable maximum-period DX/DL/DS generators of very large orders.

For each generator presented, we also compute its "max gap" ($d_{k+1}(k)$) between two adjacent parallel hyperplanes covering generated points (tuples) in $(k+1)$ dimensions. Among the maximum-period MRGs with the same order k (and the same p), an MRG with a smaller value of $d_{k+1}(k)$ is considered a better MRG under this figure of merit. Because the value of $d_{k+1}(k)$ is usually small (as it should be for a "good" generator), we scale it up by a factor of 10^5 for better presentation and easier comparison.

4.1. DX-k-s generators

Deng and Xu [5] and Deng [6] proposed DX generators as a class of portable, efficient, and maximum-period MRGs, in which the coefficients of the nonzero multipliers are the same. As a special form of (1),

1. when $\alpha_1 = 1, \alpha_k = B$, it is a DX-$k$-1 or an FMRG (Fast MRG) as considered in [27]:
$$X_i = X_{i-1} + BX_{i-k} \bmod p, \quad i \geq k; \tag{6}$$
2. when $\alpha_1 = B, \alpha_k = B$, it is a DX-$k$-2 generator with two nonzero terms of the same multiplier:
$$X_i = B(X_{i-1} + X_{i-k}) \bmod p, \quad i \geq k; \tag{7}$$
3. when $\alpha_1 = \alpha_{\lceil k/2 \rceil} = \alpha_k = B$, it is a DX-$k$-3 generator with three nonzero terms:
$$X_i = B(X_{i-1} + X_{i-\lceil k/2 \rceil} + X_{i-k}) \bmod p, \quad i \geq k; \tag{8}$$
4. when $\alpha_1 = \alpha_{\lceil k/3 \rceil} = \alpha_{\lceil 2k/3 \rceil} = \alpha_k = B$, it is a DX-$k$-4 generator with four nonzero terms:
$$X_i = B(X_{i-1} + X_{i-\lceil k/3 \rceil} + X_{i-\lceil 2k/3 \rceil} + X_{i-k}) \bmod p, \quad i \geq k. \tag{9}$$

Here the notation $\lceil x \rceil$ is the ceiling function of a number x, returning the smallest integer $\geq x$. For the class DX-k-s, s is the number of terms with coefficient B.

According to [13], a necessary (but not sufficient) condition for a "good" MRG is that the sum of squares of coefficients, $\sum_{i=1}^{k} \alpha_i^2$, should be large. Therefore, one would prefer MRGs with large $\sum_{i=1}^{k} \alpha_i^2$. For DX generators, this would ask for s and B to be as large as possible while retaining the efficiency and portability property.

For portability, Deng [6] discussed some common approaches that impose certain limits on the size of B so that exactly the same result of the multiplication can be produced with all computing platforms. In particular, Deng [6] proposed to impose a uniform upper bound for B as $B < 2^{30}$. The ISO C99 standard defines the signed integer types `long long` and `int64_t`. The former can store at least all numbers between (-2^{63}) and $(2^{63}-1)$ (including the bounds). The latter has a word size of exactly 64 bits and may not be available on all platforms. Without using 64-bit data types/operations, Deng [6] proposed a portable implementation of MRGs at the expense of slight generating inefficiency by finding an integer e such that
$$B + e \times p = C_1 \times C_2 \quad \text{and} \quad 0 < C_1, C_2 < 2^{19}. \tag{10}$$
In this case, multiplying a large B can be replaced by successive multiplications of smaller C_1 and C_2.

Using Table 1 and the efficient search algorithm GMP, it was straightforward to find DX-k-s generators. In Tables 2a–2d, we list some DX-k-s generators along with their "max gap" values $d_{k+1}(k)$ for the new (k, p)'s given in Table 1 with three possible choices: min B (smallest possible value of B for the corresponding DX generator), $B < 2^d$ ($d = 20$ for $s = 1, 2$ and $d = 19$ for $s = 3, 4$), and $B < 2^{30}$. See [6] for more details for such choices.

For DX generators listed in Tables 2a–2d, we can see that the generators with a small B (under column labeled as "(a) min B") in general have a larger value of $d_{k+1}(k)$ among the three (but with quite a few exceptions). For example, in Table 2a, for DX-15013-1 generators, the one under column (a) has the smallest multiplier $B = 1002$ and the largest value of $d_{k+1}(k)$ (99.80×10^{-5}). On the other hand, the DX generators with the largest B (under column (c)) may not always have the smallest value of $d_{k+1}(k)$. For example, in Table 2d, among the three generators listed for DX-25013-4, the smallest multiplier ($B = 35{,}304$) has the best $d_{k+1}(k)$ (1.43×10^{-5}) whereas the largest multiplier ($B = 1073{,}733{,}754$) has the worst $d_{k+1}(k)$ (4.83×10^{-5}). Also, if we classify DX-k-s in terms of s, $s = 1, 2, 3, 4$, it appears that the DX-k-4 generators in Table 2d have better $d_{k+1}(k)$ values in general, which may be due to the fact that these generators have more non-zero terms than those generators in Tables 2a–2c.

Another way to have a large sum of squares of coefficients of the recurrence equation, $\sum_{i=1}^{k} \alpha_i^2$, is to have as many nonzero terms as possible while retaining the efficiency and portability. The following DL/DS generators are such generators.

Table 2a
List of DX-k-1 with min B, $B < 2^{20}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ ($\times 10^5$).

k	(a) min B	(b) $B < 2^{20}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	8740	1046,923	1073,664,067	1	47,310	68,059	11.44	6.52	1.81
12,007	32,149	1040,102	1073,714,070	2	25,537	207,292	3.11	6.97	1.93
13,001	24,041	1034,426	1073,645,435	0	3805	282,167	4.16	2.06	3.19
14,009	14,899	1046,516	1073,730,141	0	19,899	53,959	6.71	2.66	2.74
15,013	1002	1030,728	1073,727,758	4	37,730	255,173	99.80	1.98	3.83
20,011	26,511	1012,339	1073,684,136	0	16,968	63,277	3.77	9.04	7.34
25,013	97,980	1007,372	1073,707,771	0	21,509	49,919	2.75	1.91	2.06

Table 2b
List of DX-k-2 with min B, $B < 2^{20}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ ($\times 10^5$).

k	(a) min B	(b) $B < 2^{20}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	1856	1047,362	1073,730,907	1	10,079	319,470	38.10	2.74	2.47
12,007	7648	1025,393	1073,737,306	1	22,773	139,801	9.24	4.75	2.91
13,001	3203	1032,613	1073,621,150	0	3571	300,650	22.08	4.21	2.34
14,009	60,009	1041,819	1073,738,204	0	30,851	34,804	2.00	2.20	2.13
15,013	51,475	1039,151	1073,723,417	1	6977	460,400	2.28	3.10	1.93
20,011	16,843	863,441	1073,725,998	1	40,015	79,847	4.20	5.57	4.04
25,013	39,434	969,323	1073,692,717	0	20,197	53,161	1.84	2.15	2.56

Table 2c
List of DX-k-3 with min B, $B < 2^{19}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ ($\times 10^5$).

k	(a) min B	(b) $B < 2^{19}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	1941	484,198	1073,725,785	1	51,952	61,979	29.74	11.14	2.77
12,007	5864	510,887	1073,740,063	0	7601	141,263	9.84	4.56	1.68
13,001	20,117	496,756	1073,708,125	0	4375	245,419	2.87	2.99	8.07
14,009	46,683	380,859	1073,736,613	8	45,641	399,829	2.17	1.69	1.69
15,013	7829	510,991	1073,642,398	6	41,953	331,432	7.38	5.64	2.36
20,011	31,616	514,303	1073,727,681	0	5869	182,949	1.83	3.08	1.88
25,013	79,117	492,640	1073,684,077	2	15,971	334,705	2.07	2.80	1.76

Table 2d
List of DX-k-4 with min B, $B < 2^{19}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ ($\times 10^5$).

k	(a) min B	(b) $B < 2^{19}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	16,846	501,164	1073,698,910	0	3511	305,810	2.97	2.58	1.46
12,007	13,992	521,891	1073,718,675	0	16,425	65,371	3.57	1.65	2.74
13,001	95,216	497,838	1073,737,394	4	89,766	107,641	1.83	3.12	1.54
14,009	76,837	467,248	1073,685,011	0	7889	136,099	1.61	1.88	1.99
15,013	18,597	491,619	1073,715,927	1	32,990	97,369	2.69	2.45	2.30
20,011	99,246	481,952	1073,717,310	1	46,819	68,243	1.54	1.89	1.60
25,013	35,304	490,509	1073,733,754	0	8102	132,527	1.43	1.90	4.83

4.2. DL/DS generators

Li [28] and Deng et al. [7] considered a class of generators called the DL-k generator, with $\alpha_i = B$ for $i = 1, 2, \ldots, k$, as

$$X_i = B(X_{i-1} + X_{i-2} + \cdots + X_{i-k}) \bmod p, \quad i \geq k. \tag{11}$$

Such DL generators can be implemented efficiently by

$$X_i = X_{i-1} + B(X_{i-1} - X_{i-(k+1)}) \bmod p, \quad i \geq k+1. \tag{12}$$

Deng et al. [7] also considered another class of generators with many nonzero coefficients, called the DS generator, in which $\alpha_i = B$ for all $i \in \{1, 2, \ldots, k\}$ but $i \neq d$ and $\alpha_d = 0$. Specifically, the DS generator has exactly one zero coefficient at the d-th term:

$$X_i = B \sum_{j=1,\, j\neq d}^{k} X_{i-j} \bmod p, \tag{13}$$

which can be efficiently implemented by

$$X_i = X_{i-1} + B(X_{i-1} - X_{i-d} + X_{i-d-1} - X_{i-k-1}) \bmod p, \quad i \geq k+1. \tag{14}$$

Table 3
List of DL-k with min B, $B < 2^{20}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ $(\times 10^5)$.

k	(a) min B	(b) $B < 2^{20}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	974	1047,354	1073,740,420	0	12,340	87,013	0.98	0.37	1.19
12,007	71,809	1019,967	1073,701,152	1	33,091	96,209	0.30	0.24	0.30
13,001	45,877	1046,701	1073,737,383	6	87,751	159,051	0.21	0.29	0.23
14,009	3033	1045,015	1073,731,310	6	32,768	425,869	0.28	0.19	0.35
15,013	27,470	1022,968	1073,682,207	2	14,639	365,507	0.28	0.20	0.20
20,011	9856	957,798	1073,740,275	1	14,978	213,319	0.22	0.19	0.26
25,013	77,849	1030,614	1073,632,412	0	3524	304,663	0.36	0.21	0.34

Table 4
List of DS-k with min B, $B < 2^{19}$, and $B < 2^{30}$ and their "max gap" $d_{k+1}(k)$ $(\times 10^5)$.

k	(a) min B	(b) $B < 2^{19}$	(c) $B < 2^{30}$	e	C_1	C_2	(a)	(b)	(c)
11,003	2,970	499,001	1073,738,082	0	3,642	294,821	0.32	0.24	0.25
12,007	30,905	449,500	1073,718,847	0	4,483	239,509	0.20	0.43	0.27
13,001	54,823	523,735	1073,719,552	2	13,873	386,946	0.21	0.20	0.24
14,009	17,750	511,417	1073,726,903	0	2221	483,443	0.22	0.29	0.36
15,013	12,820	489,924	1073,729,060	0	7039	152,540	0.20	0.20	0.45
20,011	8,182	523,178	1073,718,965	0	26,417	40,645	0.19	0.30	0.64
25,013	48,418	485,316	1073,732,301	10	142,831	157,061	0.25	0.29	0.17

The parameter d of the zero-coefficient index can be chosen arbitrarily. Following [7], we refer to the case of $d = \lceil k/2 \rceil$ as the DS-k generators.

In Tables 3 and 4, we list some DL-k and DS-k generators, respectively, for the new (k, p)'s given in Table 1 with min B, $B < 2^{20}$ or 2^{19}, and $B < 2^{30}$ along with their "max gap" values $d_{k+1}(k)(\times 10^5)$. We consider $B < 2^{20}$ for a double precision implementation of DL-k because its efficient recursive implementation (12) has a similar structure as that of the DX-k-2 generator (7). Similarly, $B < 2^{19}$ is considered for the DS generator because the structure of the implementation (14) is similar to that of the DX-k-4 generator (9).

For DL/DS generators listed in Tables 3 and 4, it is noted that they all have better $d_{k+1}(k)$ values than those for the DX generators listed Tables 2a–2d. Except for the smallest order ($k = 11{,}003$), all the listed DL/DS-k generators have $d_{k+1}(k) < 10^{-5}$ whereas all the DX generators have $d_{k+1}(k) > 10^{-5}$. This effect confirms nicely the advantageous design of having so many nonzero terms. Furthermore, it seems that the size of the multiplier B has little or no effect on the size of the $d_{k+1}(k)$ value. Comparing the DL and DS generators, they have similar performances in terms of the max gap $d_{k+1}(k)$.

In summary, the choice of the classes generators (DX, DL or DS) appears to have a major effect on the size of $d_{k+1}(k)$. On average, DS generators tend to have the best performance among the three classes (but just slightly better than DL generators). The worst performing group is DX generators.

4.3. *Empirical evaluations*

We evaluate these generators with the stringent empirical tests in the *Crush* battery of the TestU01 test suite. The Crush battery has 144 tests and its running time is about 1.5 h to test each generator. See [8] for more details. It appears that the conclusions of the empirical evaluations for various versions of TestU01 (v1.0, v1.1, or v1.2) do not vary much. In this paper, we report the results from TestU01 v1.1.

There are 21 (number of k's $\times$ number of B's) generators for each subclass of the DX-k-s (for $s = 1, 2, 3, 4$) in Tables 2a–2d, the DL generators in Table 3, and the DS generators in Table 4. In total, there are 126 generators listed in Tables 3, 4 and 2a–2d. For each generator, we apply the Crush battery of tests to five sets of random numbers generated with five different starting seed vectors. Each seed vector consists of k initial seeds generated by an LCG: $X_i = BX_{i-1} \bmod p$, where the multiplier B and modulus p are the same as that of the MRG under testing. The five seeds used in this study for the LCG are 1, 12, 123, 1234, and 12,345. In total, the required computing time for 630 ($= 126 \times 5$) Crush testings (each takes about 1.5 h) is almost 40 days. We obtain 15,120 ($= 144 \times 21 \times 5$) p-values for each class of the generators listed. The size of a p-value represents the probability of observing a test statistic more extreme than the one observed when the null hypothesis is true. The smaller the p-value is, the more significant the test result gets, which usually indicates that the generator fails the particular test more severely. On the other hand, when the p-value is too close to 1, it is considered as "too good to be truly random". See, for example, [8] for a more detailed discussion. The numbers of tests with p-values less than α or larger than $1 - \alpha$ for $\alpha = 10^{-3}, 10^{-4}, 10^{-5}$, and 10^{-6} are tabulated in Table 5 for the DX generators and in Table 6 for the DL and DS generators under testing.

As we can see from Tables 5 and 6, none of these tests produces a p-value smaller than 10^{-6} or larger than $1–10^{-5}$. As mentioned earlier, a small p-value means the test statistic is too far from its theoretical value and a p-value too close to 1 means the test statistic is too close to its theoretical value to be considered "truly random". In Table 5 (for DX generators), the proportions of tests producing p-values below 10^{-3}, 10^{-4}, and 10^{-5} are 0.00121, 0.00020, and 0.00002, respectively,

Table 5
Results of Crush tests on DX-k-s with $s = 1, 2, 3, 4$ and five starting seeds.

RNG	p-value	$< 10^{-3}$	$< 10^{-4}$	$< 10^{-5}$	$< 10^{-6}$	$> 1–10^{-3}$	$> 1–10^{-4}$	$> 1–10^{-5}$
DX-k-s (15,120 p-values each)								
DX-k-1	Count	24	3	0	0	18	3	0
Table 2a	Proportion	0.00159	0.00020	0	0	0.00119	0.00020	0
DX-k-2	Count	10	1	0	0	9	2	0
Table 2b	Proportion	0.00066	0.00007	0	0	0.00060	0.00013	0
DX-k-3	Count	17	3	0	0	12	2	0
Table 2c	Proportion	0.00112	0.00020	0	0	0.00079	0.00013	0
DX-k-4	Count	22	5	1	0	14	1	0
Table 2d	Proportion	0.00146	0.00033	0.00006	0	0.00093	0.00007	0
DX-k (60,480 (= 15,120 × 4) p-values in total)								
DX-k	Count	73	12	1	0	53	8	0
Tables 2a–2d	Proportion	0.00121	0.00020	0.00002	0	0.00088	0.00013	0

Table 6
Results of Crush tests on DL and DS generators with five starting seeds.

RNG	p-value	$< 10^{-3}$	$< 10^{-4}$	$< 10^{-5}$	$< 10^{-6}$	$> 1–10^{-3}$	$> 1–10^{-4}$	$> 1–10^{-5}$
DS-k (15,120 p-values each)								
DL-k	Count	15	2	0	0	16	0	0
Table 3	Proportion	0.00099	0.00013	0	0	0.00106	0	0
DL-k (15,120 p-values each)								
DS-k	Count	17	1	0	0	15	0	0
Table 4	Proportion	0.00112	0.00007	0	0	0.00099	0	0
DL-k and DS-k (30,240 (= 15,120 × 2) p-values in total)								
DL+DS	Count	32	3	0	0	31	0	0
Tables 3 and 4	Proportion	0.00106	0.00010	0	0	0.00103	0	0

which are in the same order as their corresponding nominal values (i.e., 10^{-3}, 10^{-4}, and 10^{-5} by considering p-values are uniformly distributed in (0,1)). Table 6 shows even better test results for the DL and DS generators: the proportions of tests producing p-values below 10^{-3}, 10^{-4}, and 10^{-5} are 0.00106, 0.00010, and 0.00000, respectively. In addition, none of these 90,720 tests produces a p-value very close to 0 or 1. Thus, we can conclude that each of the generators listed in Tables 3, 4 and 2a–2d has passed the Crush battery of TestU01.

5. Conclusion

In this study, we utilized some classical number theory results to develop an efficient method for the computer search of very-large-order MRGs. Having conducted an extensive computer search, we were able to identify many special-class MRGs that are portable and efficient. All of the generators presented in this paper have the equi-distribution property in very high dimensions, extremely long periods, and superior empirical performances. The largest order of the MRGs found in this study is 25,013, which has a period length approximately $10^{233,361}$, much larger than the most practical applications would need at the present time. The property of equi-distribution in dimensions up to 25,013 can be an appealing feature of the proposed MRGs especially for very-large-scale simulation studies now and in the future. If the size of memory to store $k = 25{,}013$ current state variates is a concern, one can certainly consider a smaller k.

Acknowledgment

This research was partially supported by the National Science Council in Taiwan, ROC, Grant No. NSC96-2118-M-009-004-MY2 and NSC97-2118-M-009-002-MY2. This work was done while the first author was visiting the National Center for Theoretical Sciences and Institute of Statistics, National Chiao Tung University, Hsinchu, Taiwan. We also acknowledge the usage of the high speed computing facility provided by the University of Memphis in USA for this research.

References

[1] J.D. Alanen, D.E. Knuth, Tables of finite fields, Sankhyā, Series A 26 (1964) 305–328.
[2] D.E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd ed., Addison-Wesley, Reading, MA, 1998.
[3] P. L'Ecuyer, F. Blouin, R. Couture, A search for good multiple recursive linear random number generators, ACM Transactions on Modeling and Computer Simulation 3 (1993) 87–98.
[4] L.Y. Deng, Generalized Mersenne prime number and its application to random number generation, in: H. Niederreiter (Ed.), Monte Carlo and Quasi-Monte Carlo Methods 2002, Springer-Verlag, 2004, pp. 167–180.
[5] L.Y. Deng, H. Xu, A system of high-dimensional, efficient, long-cycle and portable uniform random number generators, ACM Transactions on Modeling and Computer Simulation 13 (4) (2003) 299–309.

[6] L.Y. Deng, Efficient and portable multiple recursive generators of large order, ACM Transactions on Modeling and Computer Simulation 15 (1) (2005) 1–13.
[7] L.Y. Deng, H. Li, J.-J.H. Shiau, G.H. Tsai, Design and implementation of efficient and portable multiple recursive generators with few zero coefficients, in: S. Heinrich, A. Keller, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer-Verlag, 2008, pp. 263–273.
[8] P. L'Ecuyer, R. Simard, TestU01: a C library for empirical testing of random number generators, ACM Transactions on Mathematical Software 33 (22) (2007) 1–40.
[9] D.H. Lehmer, Mathematical methods in large-scale computing units, in: Proceedings of the Second Symposium on Large Scale Digital Computing Machinery, Harvard University Press, Cambridge, MA, 1951, pp. 141–146.
[10] L.Y. Deng, R. Guo, D.K.J. Lin, F. Bai, Improving random number generators in the Monte Carlo simulations via twisting and combining, Computer Physics Communications 178 (2008) 401–408.
[11] R. Lidl, H. Niederreiter, Introduction to Finite Fields and their Applications, Revised edition, Cambridge University Press, Cambridge, MA, 1994.
[12] G. Marsaglia, Random numbers fall mainly in the planes, Proceedings of the National Academy of Sciences 61 (1968) 25–28.
[13] P. L'Ecuyer, Bad lattice structures for vectors of non-successive values produced by some linear recurrences, INFORMS Journal on Computing 9 (1997) 57–60.
[14] M. Matsumoto, T. Nishimura, Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, ACM Transactions on Modeling and Computer Simulation 8 (1) (1998) 3–30.
[15] P. L'Ecuyer, Tables of linear congruential generators of different sizes and good lattice structure, Mathematics of Computation 68 (225) (1999) 249–260.
[16] C. Kao, H.C. Tang, Upper bounds in spectral test for multiple recursive random number generators with missing terms, Computers and Mathematical Applications 33 (1997) 113–118.
[17] G. Fishman, L.R. Moore, An exhaustive analysis of multiplicative congruential random number generators with modulus 2^{31}–1, SIAM Journal on Scientific and Statistical Computing 7 (1986) 24–45.
[18] P. L'Ecuyer, R. Couture, An implementation of the lattice and spectral tests for multiple recursive linear random number generators, INFORMS Journal on Computing 9 (2) (1997) 206–217.
[19] H.C. Williams, E. Seah, Some primes of the form $(a^n - 1)/(a - 1)$, Mathematics of Computation 33 (1979) 1337–1342.
[20] J. Brillhart, D.H. Lehmer, J.L. Selfridge, B. Tuckerman, S.S. Wagstaff Jr., Factorizations of $b^n \pm 1$, $b = 2, 3, 5, 6, 7, 10, 11, 12$ Up to High Powers, third edition, American Mathematical Society, 2002.
[21] P. L'Ecuyer, Good parameter and implementations for combined multiple recursive random number generators, Operations Research 47 (1999) 159–164.
[22] L.Y. Deng, Issues on computer search for large order multiple recursive generators, in: S. Heinrich, A. Keller, H. Niederreiter (Eds.), Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer-Verlag, 2008, pp. 251–261.
[23] H.C. Williams, Edouard Lucas and Primality Testing, John Wiley, New York, 1998.
[24] L.Y. Deng, J.-J.H. Shiau, H.H.-S. Lu, Large-order multiple recursive generators with modulus $2^{31} - 1$, INFORMS Journal on Computing (2011) http://dx.doi.org/10.1287/ijoc.1110.0477.
[25] H. Riesel, Prime Numbers and Computer Methods for Factorization, second edition, Birkhäuser, Boston, 1994.
[26] R. Crandall, C. Pomerance, Prime Numbers—A Computational Perspective, Springer-Verlag, New York, NY, 2000.
[27] L.Y. Deng, D.K.J. Lin, Random number generation for the new century, American Statistician 54 (2000) 145–150.
[28] H. Li, A System of Efficient and Portable Multiple Recursive Generators of Large Order, Ph.D. Dissertation, University of Memphis, 2005.