Designing Asymptotically Random GFSR Sequences

Hozumi Morohosi Masanori Fushimi [*]

Abstract

Tootill et al. proposed the concept of an asymptotically random Tausworthe sequence, and gave an example of such sequence found by chance. However, no systematic way of finding such sequences has been proposed by now. In this paper we will give an algorithm for designing GFSR sequences which satisfy the asymptotic randomness approximately. Our algorithm is based on repeated applications of the algorithm proposed by Fushimi. The sequence designed by the present algorithm have an added merit that their decimated sequences are also approximately asymptotically random. Some numerical examples will be shown.

1 Introduction

Since von Neumann used the middle-square method for generating pseudorandom numbers in Monte Carlo method more than half a century ago, hundreds of papers have been published on practical methods of random number generation. On the other hand, not a few papers have also been published on the theoretical aspect of random number generation, i.e., the trial to give an answer to the fundamental question, "what is a random sequence ?," and some examples can be found in [6]. However, there has been almost no connection between these two fields of research. In this paper we will present an example of a trial to bridge the gap between the two fields.

One of the fundamental concepts for defining the random sequence is "k-distribution" [6], which is also called equidistribution of order k. An l-bit number sequence $\{x_t; t = 0, 1, 2, \ldots\}$ is said to be k-distributed if the sequence of k-tuples $(x_t, x_{t+1}, \ldots, x_{t+k-1})$, $t = 0, 1, 2, \ldots$, takes each possible value with the equal probability $1/2^{kl}$.

Since a k-tuple of l-bit numbers can take 2^{kl} different values, the following inequality holds for a sequence with period T:

$$2^{kl} \leq T.$$

Hence the maximum order of equidistribution is $\lfloor \log_2 T/l \rfloor$. If the period is the same, the shorter the bit length becomes, the higher the maximum possible order of equidistribution. This shows that for an l-bit sequence the leading l'-bit, $l' < l$, can attain a higher order of equidistribution than $\lfloor \log_2 T/l \rfloor$. Based on this observation, Tootill et al. [11] proposed the notion of "asymptotic randomness," of which definition will be given in the following, and gave an example of asymptotically random sequence. But they did not refer to any method for designing such a sequence.

[*]Department of Mathematical Engineering and Information Physics, Graduate School of Engineering, University of Tokyo, 7–3–1, Hongo, Bunkyo-ku, Tokyo, 113–8656, Japan. e-mail: morohosi@misojiro.t.u-tokyo.ac.jp, fushimi@misojiro.t.u-tokyo.ac.jp

On the other hand, Kolmogorov [7] defined a finite random sequence as the sequence whose appropriately long subsequences have the same frequency distribution over the possible outcomes as the original sequence. Also for the practical use, a pseudorandom number sequence is often required to have a property that its subsequences also satisfy a certain condition of randomness. For example, when a simulation requires n random variables at a time, the n decimated sequences $\{x_{nt+i}; t = 0, 1, 2, \ldots\}$, $0 \le i \le n-1$, may be used in each step. In this case, it is desirable that these n subsequences are identically distributed. Although many kind of rules for selecting subsequences can be thought of, we consider in this paper a simple rule that every nth term is selected. Since n can be different from one simulation to another, it is desirable that the above property holds for any n in a certain set of positive integers N. Fushimi [3] proposed a method for designing a random sequence (based on a GFSR sequence) whose decimated sequences for any n in N are k-distributed.

We extend Fushimi's method to give an approximate algorithm for designing an asymptotically random number generator. Using our algorithm we can obtain a random number generator such that not only the original sequence but also its decimated sequences for any n in a given positive integer set N are approximately asymptotically random.

In the following, some definitions and properties about GFSR sequences are given in Sect. 2. In Sect. 3 an algorithm for designing an approximately asymptotically random number generator is described. Some numerical examples are presented in Sect. 4.

2 Concept of Asymptotic Randomness

Let $\{a_t\}$ be a (0, 1) sequence satisfying the pth order linear recurrence relation

$$a_t = c_1 a_{t-1} + c_2 a_{t-2} + \cdots + c_p a_{t-p} \pmod 2 \qquad (1)$$

whose characteristic polynomial

$$f(z) = 1 + c_1 z + \cdots + c_p z^p, \quad c_p = 1, \qquad (2)$$

is primitive over the Galois field GF(2), given any initial values $(a_1, \ldots, a_p) \ne (0, \ldots, 0)$. The sequence $\{a_t\}$ is periodic, and the period is $2^p - 1$.

Lewis and Payne [8] proposed to build a sequence $\{y_t\}$ of l-bit binary numbers, which they called a GFSR sequence, from $\{a_t\}$ as follows.

$$y_t = 0.a_t a_{t+\tau} a_{t+2\tau} \ldots a_{t+(l-1)\tau} \quad \text{(in base 2)}.$$

In their paper the characteristic polynomial is chosen as a primitive trinomial $f(x) = 1 + x^q + x^p$ over GF(2), and τ is a "judiciously selected" constant. GFSR is notable as a method for generating the sequence very fast because it satisfies the recurrence relation $y_t = y_{t-q} \oplus y_{t-p}$ so that one random number is generated with only one bitwise-exclusive-or operation $\oplus$.

The notion of k-distribution is a criterion of randomness. The following is the precise definition of k-distribution for the GFSR sequence. Define $[\cdot]_\lambda$ as the truncation operator to the λth fractional bit in the binary notation, i.e., when $x = 0.a_1 a_2 \ldots a_\lambda \ldots a_l$ in base 2, $[x]_\lambda = 0.a_1 a_2 \ldots a_\lambda$.

Definition 1 *An l-bit GFSR sequence $\{y_t\}$ with the period $2^p - 1$ is k-distributed to λ-bit accuracy $(1 \le \lambda \le l)$ if for each k-tuple of λ-bit binary fractional numbers $(w_1, \ldots, w_k)$ which are not all zero, $\{y_t\}$ satisfies*

$$\Pr([y_t]_\lambda = w_1, [y_{t+1}]_\lambda = w_2, \ldots, [y_{t+k-1}]_\lambda = w_k) = \frac{2^{p-k\lambda}}{2^p - 1},$$

and

$$\Pr([y_t]_\lambda = 0, [y_{t+1}]_\lambda = 0, \ldots, [y_{t+k-1}]_\lambda = 0) = \frac{2^{p-k\lambda} - 1}{2^p - 1},$$

where "probability" Pr *means the ratio of the number of events to the period.*

In the above definition the difference of the probability for the set of all zero tuple is due to the fact that the p-tuple of zeroes does not occur in $\{a_t\}$. The next theorem [4] is fundamental to our investigation.

Theorem 1 *An l-bit GFSR sequence $\{y_t\}$ is k-distributed to λ-bit accuracy if and only if all the sequences $\{a_n\}$ constituting the $k\lambda$ bits of $[y_t]_\lambda, \ldots, [y_{t+k-1}]_\lambda$ of the sequence are linearly independent over* GF(2).

Here the sequences $\{a_n^{(1)}\}, \ldots, \{a_n^{(m)}\}$ are called "linearly independent" if there exist elements $u_1, \ldots, u_m \in \{0, 1\}$, not all zero, such that $u_1 a_n^{(1)} + \cdots + u_m a_n^{(m)} = 0$ for any $n \geq 1$.

In practice we check the k-distribution property of the GFSR sequence via the equivalent Tausworthe sequence [9]. The reason why we use Tausworthe sequence will be shown in the following. Tausworthe sequence $\{x_t\}$ is constructed by the sequence $\{a_n\}$ (1) as follows:

$$x_t = 0.a_{\sigma t+1} a_{\sigma t+2} \ldots a_{\sigma t+l} \quad \text{(in base 2)}, \tag{3}$$

where σ is a constant which is relatively prime to $2^p - 1$, and $\sigma \geq l$.

These two sequences, GFSR sequence and Tausworthe sequence, which seem to be completely different, have a very close relation with each other as follows. Suppose that the condition $\sigma \geq l$ for $\{x_t\}$ is removed and that the characteristic polynomial for $\{y_t\}$ is not necessarily trinomial. We use the notations $\{x_t(f;\sigma)\}$ and $\{y_t(f;\tau)\}$ to specify the characteristic polynomials and the parameters of the sequences, and the notation $\{x_t\} \simeq \{y_t\}$ to denote the equivalence relation in the sense that two sequences $\{x_t\}$ and $\{y_t\}$ are the same except the starting points. Let σ^{-1} be the inverse of σ modulo $2^p - 1$. If n is relatively prime to the period $2^p - 1$, the sequence obtained by selecting every nth term of $\{a_t\}$, which we call a properly decimated sequence, also has the period $2^p - 1$. Let f_n be the characteristic polynomial of the properly decimated sequence. The following theorem may be easily proved by the readers familiar with the theory of finite fields (for the detailed relation between GFSR sequences and Tausworthe sequences, refer to [10]).

Theorem 2 (cf. [5]) *For a primitive polynomial f of degree p and positive integers σ, τ that are relatively prime to $2^p - 1$, we have*

$$\begin{aligned} \{x_t(f;\sigma)\} &\simeq \{y_t(f_\sigma;\sigma^{-1})\}, \\ \{y_t(f;\tau)\} &\simeq \{x_t(f_\tau;\tau^{-1})\}. \end{aligned}$$

When the parameter σ is a power of 2, we can show that $f_\sigma = f$, $\sigma^{-1} = 2^p/2^l$ in this case.

We develop a procedure to check the k-distribution property of the given Tausworthe sequence. By using the recurrence relation (1) each $a_{\sigma(t+i)+j}$ can be expressed as a linear combination of the successive p terms $a_t, \ldots, a_{t+p-1}$ in the form

$$a_{\sigma(t+i)+j} = \sum_{n=0}^{p-1} e_{jn}^{(i)} a_{t+n} \quad (i = 0, \ldots, k-1;\ j = 1, \ldots, \lambda). \tag{4}$$

Thus, the p-dimensional weight vector

$$\mathbf{e}_j^{(i)} = (e_{j0}^{(i)}, \ldots, e_{j,p-1}^{(i)})^{\mathrm{T}} \quad (i = 0, \ldots, k-1;\ j = 1, \ldots, \lambda) \tag{5}$$

can be associated with each bit sequence $a_{\sigma(t+i)+j}$. Clearly the bit sequences are linearly independent if and only if the associated vectors are linearly independent over GF(2). Since weight vectors $\mathbf{e}_j^{(i)}$ do not depend on the parameter t, we can fix the value of t to zero to consider the k-distribution property of the sequence.

The same argument applies to the sequence $\{a_{t+i+\tau(j-1)}\}$ constituting the GFSR sequence. On the other hand, the parameter τ determines the autocorrelation of the GFSR sequence, and it is desirable to make the value of τ as large as possible. When τ is large, however, we must use the recurrence relation (1) many times to obtain the weight vector $\mathbf{e}_j^{(i)}$ associated with $\{a_{t+i+\tau(j-1)}\}$. This is the reason why we use the Tausworthe sequence equivalent to the given GFSR sequence.

It is noted that the procedure is also applicable to a properly decimated sequence $\{x_{nt}\}$. Since the maximum number of linearly independent vectors does not exceed their dimension, the maximum possible order κ of equidistribution to λ bit accuracy of a Tausworthe sequence based on the primitive polynomial of degree p satisfies the following relation.

$$\kappa = \lfloor p/\lambda \rfloor. \tag{6}$$

Here $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. The equation (6) implies that the maximum order of equidistribution of a Tausworthe sequence truncated to λ-bit accuracy ($1 \leq \lambda \leq l$) is $\lfloor p/\lambda \rfloor$. If the sequence attains this maximum order for any λ ($1 \leq \lambda \leq l$) , it may be considered as an optimal sequence in the sense of k-distribution. Such a sequence is called an asymptotically random sequence [11].

Given a Tausworthe sequence $\{x_t\}$ and a positive integer n relatively prime to the period $2^p - 1$, a sequence $\{x_{nt}\}$ obtained by selecting every nth term from the Tausworthe sequence also has the period $2^p - 1$. We will call such a sequence $\{x_{nt}\}$ a properly decimated Tausworthe sequence.

In the following sections, we formulate a design problem for a given Tausworthe sequence as well as its properly decimated sequences to have asymptotic randomness, and consider an approximation algorithm to solve the problem.

3 An Algorithm for Realizing Approximate Asymptotic Randomness

3.1 Improvement of the Order of Equidistribution by Bits Rearrangement

From Theorem 1, given the characteristic polynomial (2) and parameter σ, the order of equidistribution to λ-bit accuracy of the Tausworthe sequence (3) is completely determined. An exhaustive search is needed to find a primitive polynomial and a parameter σ that yield the asymptotic randomness for the generated sequence as well as its decimated sequences. It is impossible from a practical point of view.

An alternative approach is a generalization of Tausworthe sequence [2]. Let π denote a permutation over the set $\{1, 2, \ldots, l\}$. Define the generalized Tausworthe sequence$\{x'_t\}$ based on the original Tausworthe sequence (3) as follows:

$$x'_t = 0.a_{\sigma t+\pi(1)} a_{\sigma t+\pi(2)} \cdots a_{\sigma t+\pi(l)}.$$

The goal is to find the permutation π such that for each λ, $1 \leq \lambda \leq l$, the truncated sequence $\{[x'_t]_\lambda\}$ achieves as high order of equidistribution as possible. We should point out that in this approach each truncated sequence does not necessarily attain the maximum possible order (6), because the characteristic polynomial and parameter σ is given beforehand. Since it is still difficult to find the optimal permutation for the reason that will be given below, we propose an approximation algorithm to solve the problem. The algorithm is based on Fushimi's method [3]. His method deals with the problem of finding the permutation that maximizes the number of the leading bits which are linearly independent, of the generalized Tausworthe sequence with a prescribed bit length l in consideration of the decimated sequences. The algorithm is as follows.

Algorithm S. (Selecting independent bits) We are given a primitive polynomial $f(x)$ of degree p, a parameter σ, a bit length l, the array of bit indices $L = (1, \dots, l)$, and a set of positive integers for the decimation N. Let $L(j)$ indicate the jth element of L. This algorithm finds the number of leading bits l' such that the truncated permuted sequence $\{[x'_t]_{l'}\}$ is $\lfloor p/l \rfloor$-distributed and the array of bit indices L_1 to be mapped to the leading positions. The corresponding permutation π maps L_1 to $(1, \dots, l')$ and $L_2 = L \setminus L_1$ to $(l'+1, \dots, l)$.

S1. Set $m \leftarrow \lfloor p/l \rfloor$.

S2. For each $n \in N$, calculate the weight column vector (cf. (4) and (5)) $\mathbf{e}_j^{(i)}$ $(i = 0, \dots, m-1;$ $j = 1, \dots, l)$ associated with each bit $a_{\sigma n(t+i)+L(j)}$ of $x_{n(t+i)}$, and arrange them to form the $p \times lm$ matrix

$$E_n = (\mathbf{e}_1^{(0)}, \mathbf{e}_2^{(0)}, \dots, \mathbf{e}_1^{(1)}, \mathbf{e}_2^{(1)}, \dots, \mathbf{e}_l^{(m-1)}),$$

Let $\mathcal{E}_n$ be the set of the column indices of E_n.

S3. Find all the minimal linear dependence relations among the column vectors in E_n for each $n \in N$. If there is no dependence relation, then $l' = l$, $L_1 = L$, and the algorithm terminates. Otherwise, let $\mathcal{C}_n$ be the set of such relations, i.e., $\mathcal{C}_n = \{C \subset \mathcal{E}_n |$the vectors whose indices are in C have a minimal linear dependence relation$\}$.

S4. For all $n \in N$, from each $C \in \mathcal{C}_n$ construct a 0-1 row vector $\mathbf{g} = (g_1, \dots, g_l)$ in such a way as

$$\begin{aligned} g_j &= 1, \quad \text{if there exists an integer } i \text{ such that the index of } \mathbf{e}_j^{(i)} \text{ is in } C, \\ g_j &= 0, \quad \text{otherwise,} \end{aligned}$$

cf. Fig. 1. Form a matrix G whose rows consist of $\mathbf{g}$'s corresponding to all the relations in $\mathcal{C}_n$ for all $n \in N$.

S5. Solve the following 0-1 integer linear programming problem.

$$\begin{aligned} \text{ILP:} \quad \text{minimize} \quad & z_0 = \sum_{j=1}^{l} z_j \\ \text{subject to} \quad & G\mathbf{z} \geq \mathbf{1}, \\ & z_j \in \{0, 1\} \quad (j = 1, \dots, l). \end{aligned}$$

where $\mathbf{z} = (z_1, \dots, z_l)^{\mathrm{T}}$, $\mathbf{1} = (1, \dots, 1)^{\mathrm{T}}$.

S6. If the optimal value of z_0 is $l - l'$, and a solution of ILP is such that $z_{j_1} = \cdots = z_{j_{l'}} = 1$ and the other z_j's are equal to 0, set $L_1 = (j_1, \dots, j_{l'})$ and $L_2 = L \setminus L_1$. The permutation π is given by

$$\pi = \left(\begin{array}{ccc|ccc} & L_1 & & & L_2 & \\ 1 & \dots & l' & l'+1 & \dots & l \end{array} \right).$$

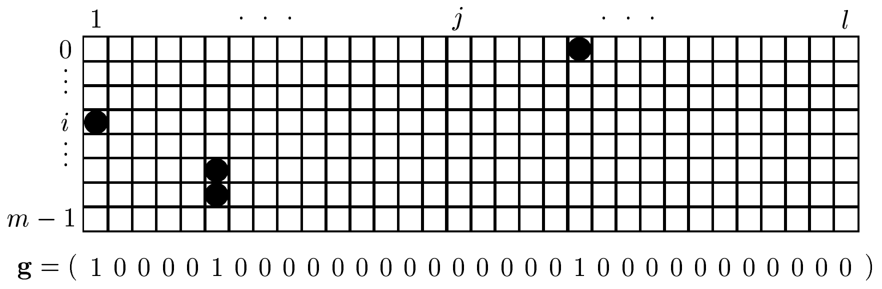
1
. . .
j
. . .
l
0
i
m − 1
g = (1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0)

Figure 1: An example of a dependence relation among the bits and the corresponding row vector. The set of the bits indicated by dots are linearly dependent.

It is to be noted that the order of vectors does not give any influence on the linear dependence or independence of these vectors. Thus, if we change the order of indices in L_1, we obtain the same order of equidistribution as far as the leading l' bits of the sequences are concerned. The basic idea of our algorithm below is to iterate Algorithm S on the successively obtained L_1's to make the given sequence approximately asymptotically random. After applying Algorithm S the truncated permuted sequence $\{[x'_t]_{l'}\}$ is $\lfloor p/l \rfloor$-distributed. If we apply Algorithm S to the sequence $\{[x'_t]_{l'}\}$, it may be possible to find a new truncated permuted sequence $\{[x''_t]_{l''}\}$ $(1 \le l'' \le l')$, which is $\lfloor p/l' \rfloor$-distributed, cf. Fig 2. The sequence obtained by such an iteration of Algorithm S may not have the maximum order of equidistribution to each accuracy. We can consider, however, it to be an approximately asymptotically random sequence.

Algorithm A. (Approximate asymptotic randomness) We are given a primitive polynomial $f(x)$ of degree p, a parameter σ, a bit length l, array of bit indices L, and a set of positive integers N for the decimation. Let trunc_s be an operator on the array L to make a new array of first s elements of L. This algorithm finds a bit permutation π by which we can realize an approximately asymptotically random sequence.

A1. Let $L = (1, \ldots, l)$ and the permutation τ be the identity permutation over L.

A2. Call Algorithm S to obtain l', L_1, L_2, and π.

A3. If $l' \le 1$, the algorithm terminates. Otherwise, if $l' < l$, set $l \leftarrow l'$ and use L_1 as the new array of bit indices L, else set $l \leftarrow l - 1$ and use $\mathrm{trunc}_{l-1}(L)$ as the new L. Set the permutation $\tau \leftarrow \pi \circ \tau$, where π is obtained in step A2. Go to step A2.

In Algorithm A, the permutations obtained at each step A2 are defined on the arrays of different sizes, so that the expression $\tau \leftarrow \pi \circ \tau$ is not well defined. By $\pi \circ \tau$ we mean the operation that we apply τ to the array of length l first, and then π to the array restricted to the leading l' bits. Since Algorithm S assures only that the sequence $\{[x'_t]_{l'}\}$ is $\lfloor p/l \rfloor$-distributed, we cannot know the order of equidistribution of the sequences $\{[x'_t]_\lambda\}$, $l' < \lambda \le l$. In this sense the obtained sequence is not asymptotically random in the exact sense, but we expect it to be approximately asymptotically random.

Despite the restricted situation we need two more approximations to implement the Algorithm A (and Algorithm S). First, at step S3 in Algorithm S we must enumerate all the minimal linear dependence relations among the column vectors in E_n. Such an enumeration is a very hard problem by the reason that we give in the next section, where a heuristic method is also presented. Second, at step S5 in Algorithm S we must solve ILP, which is an NP-complete problem known as the "set covering problem." Two approximation algorithms proposed in [3] are also used in this paper.

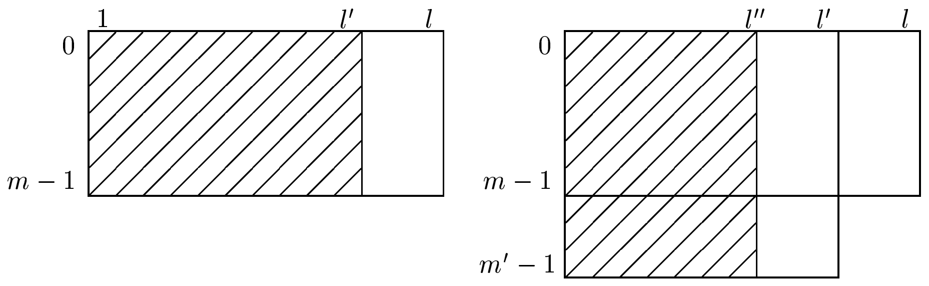

Figure 2: An illustration of approximate asymptotic randomness.
$m = \lfloor p/l \rfloor$, $m' = \lfloor p/l' \rfloor$. Each hatched area shows the set of independent bits.

3.2 Enumeration of minimal dependent sets

In this section we deal with the problem of finding all the minimal linear dependence relations among the column vectors in a given matrix E_n. Matroid theory is a fundamental tool to investigate the characteristics of independent or dependent sets over a finite set. In matroid theory, a minimal dependent set is called a *circuit*, and a maximal independent set is called a *basis*. Given a basis B and an element x which is not the member of B, there exists a unique circuit in the set $B \cup \{x\}$. This circuit is called the fundamental circuit $C(x)$ relative to the basis B. Let $\mathcal{E}_n$ be the set of all the column vectors of E_n, and $M(\mathcal{E}_n)$ be the matroid on $\mathcal{E}_n$. Our problem is to find all the circuits of the matroid $M(\mathcal{E}_n)$.

A matroid is said to be binary if it is representable over GF(2). As to the binary matroid, the next fact is well known (it is a part of theorem 2.2.1 in [1]).

Proposition 1 *A matroid is binary if and only if for every basis B and every circuit C, C is representable as follows:*

$$C = \triangle_{x \in C \setminus B} C(x), \tag{7}$$

where $C(x)$ denotes the fundamental circuit formed by an element $x \in C \setminus B$ with respect to the basis B, and $\triangle$ denotes the symmetric difference of sets [*]*; that is, C is the mod 2 sum of those circuits.*

According to the proposition, the set of all the symmetric differences $\mathcal{S} = \{\triangle_{x \in A} C(x) |\ A \subset \mathcal{E}_n \backslash B,\ A \neq \emptyset\}$ contains all the circuits of $M(\mathcal{E}_n)$; therefore we can enumerate all the circuits if we examine each element in $\mathcal{S}$ for whether it is a circuit.

Given the matrix E_n, to obtain a basis and the corresponding fundamental circuits of the matroid $M(\mathcal{E}_n)$ we can apply Gauss-Jordan elimination method to E_n. After the elimination with exchange of row and column indices, the matrix E_n may be transformed to the form

$$\left(\begin{array}{c|c} I & K \\ \hline 0 & 0 \end{array}\right),$$

where I is an identity matrix. A basis B consists of the column indices corresponding to I. For each column vector in K, the row indices of nonzero elements express the fundamental circuit relative to B.

[*] For two sets X, Y the symmetric difference of them is $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$. Since for three sets X, Y, Z, $\triangle$ satisfies the associative law $X \triangle (Y \triangle Z) = (X \triangle Y) \triangle Z$, we can define the symmetric difference for more than three sets. The RHS of expression (7) means $C(x_1) \triangle \cdots \triangle C(x_\nu)$, where we assume $C \setminus B = \{x_1, \ldots, x_\nu\}$.

Table 1: The process of selecting independent bits for the case (A) below.

l	$\lfloor p/l \rfloor$	l'	L_1
32	16	32	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
31	16	31	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
30	17	18	1 2 3 4 5 6 7 8 9 22 23 24 25 26 27 28 29 30
18	28	11	3 5 6 7 8 9 22 23 24 25 27
11	47	6	3 5 6 7 8 27
6	86	4	5 6 7 27
4	130	2	7 27
2	260	1	7

For instance, let k be an index of a column vector $\mathbf{e}_j^{(i)}$ in K and $\{k_1, \dots, k_s\}$ be the set of the row indices of nonzero elements of $\mathbf{e}_j^{(i)}$, then $\{k_1, \dots, k_s, k\}$ is a fundamental circuit relative to B.

Theoretically, we can enumerate all the circuits by the methods stated above, but it is very difficult to calculate all the symmetric differences $\mathcal{S}$ in view of computational cost, and hence we propose a heuristics. We use the set of all fundamental circuits instead of all the circuits $\mathcal{C}_n$ at step S3 in Algorithm S. It is possible that the selected bits at step S5 are linearly independent even if the enumeration of circuits is not sufficient. Since we skip the enumeration of circuits, we must check the independence of selected bits, which is easily carried out. In the next section we give some numerical experiments and demonstrate that our heuristic method is effective.

4 Numerical Experiments

First we consider the case without decimation, that is $N = \{1\}$. In this case our algorithm gives an approximately asymptotically random number generator in the original sense presented by Tootill et al.

In order to illustrate the algorithm we follow the first several iterations of the procedure step by step. We select a primitive polynomial $f(x) = x^{521} + x^{32} + 1$ over GF(2). The period of the M-sequence based on this polynomial is $2^{521} - 1$, which is a Mersenne prime number. The initialization of Algorithm A is as follows. the degree of polynomial $p = 521$, the parameter $\sigma = 32$, the bit length $l = 32$, the array of bit indices $L = (1, \dots, 32)$, and the set of positive integers for decimation $N = N_1 = \{1\}$,

In the first iteration we find there is no dependent relation among the weight vectors $\mathbf{e}_j^{(i)}$ $(i = 0, \dots, m-1;\ j = 1, \dots, l)$, and obtain $l' = 32$. The initial values in the second iteration are $l = 31$ and $L = (1, \dots, 31)$. We do not find any dependent relation either, so $l' = 31$.

The third iteration begins with $l = 30$ and $L = (1, \dots, 30)$. We find 21 fundamental circuits in E_n. They are $\{\{(0, j_1), (15, j_1), (16, j_1)\} |\ j_1 = 1, \dots, 21\}$, providing the column index for $\mathbf{e}_j^{(i)}$ is designated as (i, j). Constructing the corresponding row vectors $\mathbf{g}$, we obtain the solution of ILP: $z_0 = 12 (l' = 18)$, $L_1 = (1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 23, 24, 25, 26, 27, 28, 29, 30)$.

The fourth iteration begins with $l = 18$, $L = L_1$. The arrays of bit indices L_1 obtained in the following iterations are shown in Table 1.

We applied our algorithm to other many generators, for some of which the algorithm terminates before enough iterations (i.e. when l' is still big). We consider this is because the matrix G in ILP is incomplete. For the following four characteristic polynomials, which are all primitive polynomials over GF(2), we obtained approximately asymptotically random sequences (we have already explained the case A).

(A)	$x^{521} + x^{32} + 1$	(C)	$x^{607} + x^{273} + 1$
(B)	$x^{521} + x^{455} + x^{437} + x^{350} + 1$	(D)	$x^{607} + x^{461} + x^{307} + x^{167} + 1$

Next we take the decimation into account. The four polynomials above were used. The initialization of each case is also $l = 32$, $\sigma = 32$, and integer set for decimation $N = N_2 = \{1, 2, \ldots, 16\}$. In Table 2 we show the bit arrangement obtained by our algorithm. In the table the first column shows the polynomial, the second column shows the set for decimation. For each polynomial and decimation set, in the third column we show the bit arrangement obtained by the algorithm in the upper row and the orders of equidistribution of the leading bits in the lower row. For example, in the case that the polynomial is A without considering the decimation ($N = N_1$) the leading 18 bits of the permuted sequence is at least 17-distributed, while with considering decimation ($N = N_2$) the leading 18 bits of the permuted sequence (the permutations are different for each case) is at least 16-distributed.

Table 2: Bit arrangement obtained by the algorithm.

A	N_1	7	27	5	4	3	8	9	22	23	24	25	1	2	3	26	28	29	30	10	11	12	13	14	15	16	17	18	19	20	21	31	32
			130		80		47					28							17												16	16	16
	N_2	3	6	4	5	2	1	7	8	9	19	20	21	22	28	29	30	31	32	10	11	12	13	14	15	16	17	18	23	24	25	26	27
			130		104	57				28									16														
B	N_1	20	21	24	25	17	26	27	13	11	8	9	7	23	28	3	29	1	2	6	30	12	14	22	4	5	10	15	16	18	19	31	32
			260	173	130		86	74	57	52	47	43	40		37		32	28	26		22			17								16	16
	N_2	23	28	22	21	20	18	17	16	30	15	14	31	13	10	9	4	3	2	12	24	27	29	5	6	7	8	11	19	25	26	32	1
			173	130	104	86	74	65		57	47		43	37	34	32	30	28	23				17									16	
C	N_1	13	17	10	7	1	6	14	24	5	30	26	3	15	22	12	18	2	4	8	9	11	16	19	20	21	23	25	27	28	29	31	32
			202	151	121	101	86	75	60		55	43			37		20															19	18
	N_2	25	20	15	16	13	19	21	22	28	17	18	24	8	11	23	27	2	3	4	5	6	7	9	10	12	14	27	29	1	30	31	32
			202	151	121	67				50			37				21												18				
D	N_1	7	11	13	14	15	16	19	20	22	23	24	25	28	29	6	5	1	3	26	30	8	12	2	4	9	10	17	18	21	27	31	32
			303	202	151	121	101	86	75	67	60	55	50	46	43	37	35	33	30		27		20									19	18
	N_2	16	19	15	20	14	13	22	23	24	25	11	28	29	7	6	5	1	3	26	30	8	12	2	4	9	10	17	18	21	27	31	32
			202		151	101		86	75	67	60		50	46	40	37	35	33	30		27		20									19	18

5 Concluding Remarks

We have presented a method for designing a uniform random number generator whose decimated sequences are approximately asymptotically random. Some good generators are obtained by the method. In this paper we have dealt with Tausworthe sequence, but it is easy to transform Tausworthe sequence into GFSR sequence in terms of Theorem 2 and the following remark. Thus these sequences can be generated very fast; the speed of the generation of each random number is the same as an ordinary GFSR method, i.e., without bit permutation, because the arrangement of bits is needed only at the initialization of the random number generation program, cf. [3].

References

[1] Fournier, J. C., Binary Matroids, in *Combinatorial Geometries* (N. White, ed.) (*Encyclopedia of Mathematics and Its Applications, Vol. 29*), Cambridge University Press, Cambridge, 1987.

[2] Fushimi, M., Increasing the Orders of Equidistribution of the Leading Bits of the Tausworthe Sequence, *Inform. Process. Lett.*, Vol. 16(1983), pp. 189–192.

[3] Fushimi, M., Designing a Uniform Random Number Generator Whose Subsequences Are k-Distributed, *SIAM J. Comput.*, Vol. 17(1988), pp. 89–99.

[4] Fushimi, M. and S. Tezuka: The k-distribution of the Generalized Feedback Shift Resister Pseudorandom numbers, *Comm. ACM*, Vol. 26(1983), pp. 516–523.

[5] Fushimi, M., *Random Numbers*, University of Tokyo Press, Tokyo, 1989. (In Japanese.)

[6] Knuth, D. E., *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1981.

[7] Kolmogorov, A. N., On Tables of Random Numbers, *Sankhya*, Vol. 25A(1963), pp. 369–376.

[8] Lewis, T. G. and W. H. Payne, Generalized Feedback Shift Register Pseudorandom Number Algorithms, *J. ACM*, Vol. 20(1973), pp. 456–468.

[9] Tausworthe, R. C., Random Numbers Generated by Linear Recurrence Modulo Tow, *Math. Comp.*, Vol. 19(1965), pp. 201–209.

[10] Tezuka, S., *Uniform Random Numbers: Theory and Practice*, Kluwer, Boston, 1995.

[11] Tootill, J. P. R., W. D. Robinson, and D. J. Eagle, An Asymptotically Random Tausworthe Sequence, *J. ACM*, Vol. 20(1973), pp. 469–481.