Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 174 (2005) 165–177
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Efficiency test of pseudorandom number generators using random walks☆

Mihyun Kang*

Humboldt-Universität zu Berlin, Institut für Informatik, D-10099 Berlin, Germany

Received 8 July 2003

Abstract

Markov chain Monte Carlo methods and computer simulations usually use long sequences of random numbers generated by deterministic rules, so-called pseudorandom number generators. Their efficiency depends on the convergence rate to the stationary distribution and the quality of random numbers used for simulations. Various methods have been employed to measure the convergence rate to the stationary distribution, but the effect of random numbers has not been much discussed. We present how to test the efficiency of pseudorandom number generators using random walks.

MSC: Primary 65C10; Secondary 11K45; 60B15

Keywords: Random walks; Finite abelian groups; Pseudorandom number generators; First return times; First hitting times

1. Introduction

Many aspects of random walks have been studied: asymptotic behaviors of transition probabilities and convergence to harmonic functions [2], cutoff phenomenon [8], relation to combinatorial structures and electric networks [3,12,33], molecular recognition in polymer physics and biology [26], and the design of off or on-line randomized algorithms [6,29,34]. One would simulate random objects using random walks and predict the structural properties, when it is difficult to apply any deterministic method to analyze combinatorial structures: it is called Markov chain Monte Carlo methods

☆ This research was supported by the Deutsche Forschungsgemeinschaft through the European graduate program 'Combinatorics, Geometry, and Computation' (No. GRK 588/2).
* Tel.: +49-30-2093-3830; fax: +49-30-2093-3191.
E-mail address: kang@informatik.hu-berlin.de (M. Kang).

0377-0427/$ - see front matter
doi:10.1016/j.cam.2004.04.005

[2,21,22]. The efficiency of Monte Carlo methods depends on the convergence rate to the stationary distribution, so-called the mixing time, and random numbers used for computer simulations.

Various techniques are developed to obtain upper bounds of mixing times, using spectral properties and group representations [7,11], comparison techniques [9], canonical paths [21,22] and couplings [2,4]. They are successfully applied for random generations and countings in statistical physics and combinatorics, such as shufflings [1,17], approximate volume estimation for convex bodies [14], colorings [35] and tilings [20,28]. Moreover there have been new approaches for perfect sampling algorithms, which generate random samples distributed exactly according to the stationary distribution, such as Propp-Wilson's coupling from the past [32,36] and Fill's perfect rejection sampling algorithm [15,16].

Random numbers employed in computer simulations, cryptography, computer generated graphics and music, and other probabilistic algorithms are usually generated by deterministic rules, called pseudorandom number generators [18,27,30]. Carefully selected pseudorandom number generators improve the performance of algorithms in many applications. What are good pseudorandom number generators? Pseudorandom number generators should generate random numbers quickly and the generated random numbers should behave as if they were produced by a real random process. But random sequences generated by pseudorandom number generators may have inevitable correlations and they can lead to wrong results in simulations and other applications. Thus it is required that efficient pseudorandom number generators should pass as many statistical tests as possible.

In this paper we present random walk algorithms to test the efficiency of pseudorandom number generators. Our random walk algorithms use nearest neighbor walks on finite abelian groups such as $\mathbb{Z}_2^n = \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2$ and $\mathbb{Z}_m \times \mathbb{Z}_n$. A walker starts from a point $\vec{x}$ in the group and moves the step to one of neighbors in a randomly chosen direction. We consider the probability distributions of the first hitting time (or the first passage time) from $\vec{x}$ to $\vec{0}$, which is the minimum number of steps taken to reach $\vec{0}$ starting from $\vec{x}$ [19,24,25]: it is called the first return time when $\vec{x} = \vec{0}$. We derive the probability generating functions by adapting the arguments of Diaconis [7] based on discrete Fourier analysis so that we can efficiently compute the theoretical values of expectations, variations and even higher moments, using Taylor series expansions [13].

To empirically test the quality of a pseudorandom number generator we propose to use a random sequence $\{X_j\}_{j=0}^{\infty}$ in $\mathbb{Z}_M$ generated by a pseudorandom number generator to determine the walker's movements in computer experiments. In the algorithms on $\mathbb{Z}_2^n$ the number of possible movements is n or $n+1$ since each point of $\mathbb{Z}_2^n$ has n neighbors, and in the algorithms on $\mathbb{Z}_m \times \mathbb{Z}_n$ the number of possible movements is four or five: the additional possibility of movement arises from the possibility of staying at the current position. If we need k different movements, then we divide $[0,1]$ into k subintervals $I_i = [\frac{i-1}{k}, \frac{i}{k})$, $1 \leqslant i \leqslant k$ and move the step according to which subinterval X_j/M falls into. If the selected pseudorandom number generator is of high quality, then the walker's movements would be sufficiently random. This makes every statistical values of computer experiments close to theoretical values of expectations and variations computed by the probability generating functions.

In Section 2 we present preliminaries on random walks on finite abelian groups, group representations and its application to the probability generating functions of the first return time and the first hitting time. In Section 3 and Section 4 we propose random walk algorithms on $\mathbb{Z}_2^n$ and $\mathbb{Z}_m \times \mathbb{Z}_n$. In Section 5 and Section 6 we introduce typical pseudorandom number generators and explain the methods of testing pseudorandom number generators using random walk algorithms. In Section 7 we present some results of implementation.

2. Random walks on finite abelian groups

Let G be a finite abelian group. A character χ of G is a map $\chi : G \to \mathbb{C}$ such that $\chi(x)$ is of modulus 1 and $\chi(x+y) = \chi(x)\chi(y)$ for every $x, y \in G$. It follows that $\chi(0) = 1$ for the additive identity 0 and $\chi(-x) = \chi(x)^{-1} = \overline{\chi(x)}$. Thus χ is a group homomorphism from G into the circle group $|z| = 1$. We denote by $\widehat{G}$ the set of all characters of G which is called the dual group of G. For example, the dual group $\widehat{\mathbb{Z}_n}$ consists of characters of the form

$$\chi_k(x) = \exp\left(\frac{2\pi i k x}{n}\right)$$

for $x, k \in \mathbb{Z}_n$. Characters of a product group $G_1 \times G_2$ are products of two characters of G_1 and G_2, that is, $\chi \in \widehat{G_1 \times G_2}$ if and only if $\chi(g_1, g_2) = \chi_1(g_1)\chi_2(g_2)$ for $\chi_1 \in \widehat{G_1}, \chi_2 \in \widehat{G_2}, g_1 \in G_1, g_2 \in G_2$. For example, the dual group $\widehat{\mathbb{Z}_m \times \mathbb{Z}_n}$ consists of characters of the form

$$\chi_{\vec{k}}(\vec{x}) = \exp\left(\frac{2\pi i k_1 x_1}{m} + \frac{2\pi i k_2 x_2}{n}\right)$$

for each $\vec{k} = (k_1, k_2)$, $\vec{x} = (x_1, x_2) \in G$. For a function f defined on G, its Fourier transform $\hat{f} : \widehat{G} \to \mathbb{C}$ is defined by

$$\hat{f}(\chi) = \sum_{x \in G} f(x)\chi(x)$$

and the Fourier inversion formula is given by

$$f(x) = \frac{1}{|G|} \sum_{x \in \widehat{G}} \hat{f}(\chi)\chi(-x)$$

where $|G|$ is the number of elements in G.

Let μ be a probability distribution on a finite abelian group G. The set $\operatorname{supp}\mu = \{x \in G \mid \mu(x) > 0\}$ is called the support of μ. Suppose $\operatorname{supp}\mu$ generates G, that is, any element in G is a sum of some elements in $\operatorname{supp}\mu$. Define the convolution $\mu * \mu$ by

$$(\mu * \mu)(x) = \sum_{y \in G} \mu(x - y)\mu(y), \quad x \in G.$$

Note that $(\mu * \mu)(x)$ is the probability that a random walker arrives at x in two steps when he starts from the identity. Similarly $\mu^{*k}(x)$ is the probability that a random walker arrives at x in k steps when he starts from the identity.

Random walks on G are Markov chains with the state space G, whose transition probability from a state x to a state y is given by $\Pr(x, y) = \mu(y - x)$ for $x, y \in G$. The Markov chain related to the random walk on (G, μ) is ergodic when the support of μ generates G. Kac's Lemma [23] implies that the average of the first return time to the starting point equals the number of elements in G.

We follow the arguments in [7] to derive the formula for the generating function of first hitting times. Suppose there is an element s of G, called a sink: if a walker arrives at s, that is, he hits the sink s for the first time, then he is permanently stuck there. We consider the random walk starting from a fixed position $x(\neq s) \in G$. Let $a_k(t)$ be the probability of arriving at $t \in G$ in k steps. If $t \neq s$, then $a_k(t)$ is the chance of a walker's arriving at t in k steps without having hit the sink

s. If $t = s$ then it is the chance of first hitting the sink at time k. Let $b_k(t)$ be the probability of arriving at t in k steps under the unrestricted random walk with no sink; hence $b_k(t) = \mu^{*k}(t - x)$. Let $A(t,z), B(t,z)$ be generating functions given by

$$A(t,z) = \sum_{k=0}^{\infty} a_k(t) z^k \quad \text{and} \quad B(t,z) = \sum_{k=0}^{\infty} b_k(t) z^k.$$

The following is a special case of the result in [7].

Fact 1: (i) *For* $t \neq s$,

$$B(t,z) = A(t,z) + A(s,z)B(t - s + x, z)$$

and for $t = s$,

$$B(t,z) = A(t,z)B(x,z).$$

(ii) *For* $t \in G$,

$$B(t,z) = \frac{1}{|G|}\left(\frac{1}{1-z} + \sum_{\chi \neq 1} (x - t)[1 - \hat{\mu}(\chi)z]^{-1}\right).$$

Throughout the paper we denote by T the first hitting time or the first return time and let

$$A(z) = \sum_{k=0}^{\infty} \Pr(T = k) z^k.$$

Note that $A'(1) = \sum_{k=1}^{\infty} k \Pr(T = k) = \mathbb{E}(T)$ and $A''(1) = \sum_{k=2}^{\infty} k(k-1)\Pr(T = k) = \mathbb{E}(T^2) - \mathbb{E}(T)$ and thus the expectation and variation of T are given by

$$\mathbb{E}(T) = A'(1)$$

and

$$\sigma^2(T) = A''(1) + A'(1) - A'(1)^2.$$

Proposition 1. *The probability generating function of the first hitting time* T *from* x *to the identity* 0 *is given by*

$$A(z) = \frac{1 + (1-z)\sum_{\chi \neq 1} \chi(x)[1 - \hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}}. \tag{1}$$

Proof. Since $t = s = 0$, Fact 1 implies $A(z) = A(0,z)$ and $B(0,z) = A(0,z)B(x,z)$ where

$$B(0,z) = \frac{1}{|G|}\left(\frac{1}{1-z} + \sum_{\chi \neq 1} \chi(x)[1 - \hat{\mu}(\chi)z]^{-1}\right)$$

$$B(x,z) = \frac{1}{|G|}\left(\frac{1}{1-z} + \sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}\right). \qquad \square$$

Proposition 2. *The probability generating function of the first return time T to the identity 0 is given by*

$$A(z) = \frac{1 + (1-z)\sum_{\chi \neq 1} \hat{\mu}(\chi)[1 - \hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}} z \tag{2}$$

when the walker does not stay at the current position and

$$A(z) = \mu(0)z + \frac{(1-\mu(0)) + (1-z)\sum_{\chi \neq 1} (\hat{\mu}(\chi) - \mu(0))[1 - \hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}} z \tag{3}$$

when the walker may stay at the current position.

Proof. First we assume that the walker does not stay at the current position, i.e., $\mu(0) = 0$. The walker moves his first step to each $x \in G$ with $\mu(x) > 0$. Let $T_{0,x}$ be the first hitting time from 0 to x and $T_{x,0}$ the first hitting time from x to 0. Then for each $x \in G$ with $\mu(x) > 0$, $T_{0,x} + T_{x,0} = 1 + T_{x,0}$ since $T_{0,x} = 1$. Thus

$$\Pr(T = k) = \sum_{\mu(x)>0} \mu(x)\Pr(T_{0,x} + T_{x,0} = k) = \sum_{\mu(x)>0} \mu(x)\Pr(T_{x,0} = k-1)$$

and

$$A(z) = \sum_{k=0}^{\infty} \sum_{\mu(x)>0} \mu(x)\Pr(T_{x,0} = k-1)z^k = \sum_{\mu(x)>0} \mu(x) \sum_{k=0}^{\infty} \Pr(T_{x,0} = k) z^{k+1}.$$

Fact 1 implies $B(0,z) = A(0,z)B(x,z)$ where $A(0,z) = \sum_{k=0}^{\infty} \Pr(T_{x,0} = k)z^k$,

$$|G| \cdot B(0,z) = \frac{1}{1-z} + \sum_{\chi \neq 1} \chi(x)[1 - \hat{\mu}(\chi)z]^{-1}$$

and

$$|G| \cdot B(x,z) = \frac{1}{1-z} + \sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}.$$

Hence we obtain

$$\begin{aligned} A(z) &= \sum_{\mu(x)>0} \mu(x) \frac{1 + (1-z)\sum_{\chi \neq 1} \chi(x)[1 - \hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}} z \\ &= \frac{1 + (1-z)\sum_{\chi \neq 1} \hat{\mu}(\chi)[1 - \hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi \neq 1} [1 - \hat{\mu}(\chi)z]^{-1}} z. \end{aligned}$$

For the case $\mu(0) > 0$ we have

$$A(z) = \mu(0)z + \frac{(1-\mu(0)) + (1-z)\sum_{\chi\neq 1}(\hat{\mu}(\chi) - \mu(0))[1-\hat{\mu}(\chi)z]^{-1}}{1 + (1-z)\sum_{\chi\neq 1}[1-\hat{\mu}(\chi)z]^{-1}} z. \qquad \square$$

3. Random walk algorithms on $\mathbb{Z}_2^n$

In this section we introduce several versions of nearest neighbor walks on the hypercube. Let $\mathbb{Z}_2$ be the additive group $\{0,1\}$ and G the product group $\mathbb{Z}_2^n$. The dual group $\widehat{G}$ consists of characters of the form

$$\chi_{\vec{k}}(\vec{x}) = (-1)^{\vec{k}\cdot\vec{x}} \quad \text{for } \vec{k},\vec{x} \in G$$

where $\vec{k}\cdot\vec{x} = \sum_{i=1}^{n} k_i x_i$.

In Algorithms 3.1 and 3.2 we consider a random walk starting from $\vec{x}$ and moving to one of n neighbors with equal probability $\frac{1}{n}$. The probability distribution μ on G is given by $\mu(\vec{e}_i) = 1/n$ for $1 \leqslant i \leqslant n$. Then $\hat{\mu}(\chi_{\vec{k}}) = 1 - (2j)/n$, $\chi_{\vec{k}} \in \widehat{G}$ where j is the number of 1's in $\vec{k}$.

Algorithm 3.1. *Let T be the first return time to $\vec{0}$. A walker starts from $\vec{0}$ and moves to one of n neighbors with equal probability $\frac{1}{n}$.*

Formula (2) implies that $A(z) = zf(z)/g(z)$ where

$$f(z) = 1 + (1-z)\sum_{j=1}^{n}\binom{n}{j}\left(1-\frac{2j}{n}\right)\left[1-\left(1-\frac{2j}{n}\right)z\right]^{-1}$$

$$g(z) = 1 + (1-z)\sum_{j=1}^{n}\binom{n}{j}\left[1-\left(1-\frac{2j}{n}\right)z\right]^{-1}.$$

Algorithm 3.2. *Let T be the first hitting time from $\vec{1} = (1,\ldots,1)$ to $\vec{0}$. A walker starts from $\vec{1}$ and moves to one of n neighbors with equal probability $\frac{1}{n}$.*

Formula (1) implies that $A(z) = f(z)/g(z)$ where

$$f(z) = 1 + (1-z)\sum_{j=1}^{n}\binom{n}{j}(-1)^j\left[1-\left(1-\frac{2j}{n}\right)z\right]^{-1},$$

$$g(z) = 1 + (1-z)\sum_{j=1}^{n}\binom{n}{j}\left[1-\left(1-\frac{2j}{n}\right)z\right]^{-1}.$$

In Algorithms 3.3 and 3.4 we consider a random walk starting from $\vec{x}$ and staying at the current position or moving to one of n neighbors with equal probability. The probability distribution μ is

given by $\mu(\vec{0})=\mu(\vec{e}_i)=\frac{1}{n+1}$, $1\leqslant i\leqslant n$. Then $\hat{\mu}(\chi_{\vec{k}})=1-(2j)/(n+1)$, $\chi_{\vec{k}}\in\widehat{G}$ where j is the number of 1's in $\vec{k}$.

Algorithm 3.3. *Let T be the first return time to $\vec{0}$. A walker starts from $\vec{0}$ and stays at the current position or moves to one of n neighbors with equal probability $\frac{1}{n+1}$.*

Formula (3) implies $A(z)=z/(n+1)+zf(z)/g(z)$ where

$$f(z)=\frac{n}{n+1}+(1-z)\sum_{j=1}^{n}\binom{n}{j}\left(1-\frac{2j+1}{n+1}\right)\left[1-\left(1-\frac{2j}{n+1}\right)z\right]^{-1}$$

$$g(z)=1+(1-z)\sum_{j=1}^{n}\binom{n}{j}\left[1-\left(1-\frac{2j}{n+1}\right)z\right]^{-1}.$$

Algorithm 3.4. *Let T be the first hitting time from $\vec{1}=(1,\ldots,1)$ to $\vec{0}$. A walker starts from $\vec{1}$ and stays at the current position or moves to one of n neighbors with equal probability $\frac{1}{n+1}$.*

Formula (1) implies $A(z)=f(z)/g(z)$ where

$$f(z)=1+(1-z)\sum_{j=1}^{n}\binom{n}{j}(-1)^j\left[1-\left(1-\frac{2j}{n+1}\right)z\right]^{-1}$$

$$g(z)=1+(1-z)\sum_{j=1}^{n}\binom{n}{j}\left[1-\left(1-\frac{2j}{n+1}\right)z\right]^{-1}.$$

4. Random walk algorithms on $\mathbb{Z}_m\times\mathbb{Z}_n$

In this section we consider versions of nearest neighbor walks on the discrete torus. Let $\mathbb{Z}_n$ be the additive group $\{0,1,\ldots,n-1\}$ and G the product group $\mathbb{Z}_m\times\mathbb{Z}_n$. The characters of G are of the form

$$\chi_{\vec{k}}(\vec{x})=\exp\left(\frac{2\pi ik_1x_1}{m}+\frac{2\pi ik_2x_2}{n}\right)$$

for each $\vec{k}=(k_1,k_2)$, $\vec{x}=(x_1,x_2)\in G$. For the notational convenience, define

$$C_{k_1,k_2}=\frac{1}{2}\cos\left(\frac{2\pi k_1}{m}\right)+\frac{1}{2}\cos\left(\frac{2\pi k_2}{n}\right).$$

In Algorithms 4.1 and 4.2 we consider a random walk starting from $\vec{x}$ and moving to one of four neighbors with equal probability. The probability distribution μ is given by $\mu(1,0)=\mu(-1,0)=\mu(0,1)=\mu(0,-1)=\frac{1}{4}$. Then $\hat{\mu}(\chi_{\vec{k}})=\sum_{\vec{x}\in G}\mu(\vec{x})\chi_{\vec{k}}(\vec{x})=C_{k_1,k_2}$ for $\vec{k}=(k_1,k_2)$.

Algorithm 4.1. *Let T be the first return time to $\vec{0}$. A walker starts from $\vec{0}$ and moves right, left, up or down from a current position with equal probability.*

Formula (2) implies $A(z) = zf(z)/g(z)$ where

$$f(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} C_{k_1,k_2}(1 - C_{k_1,k_2}z)^{-1}$$

$$g(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} (1 - C_{k_1,k_2}z)^{-1}.$$

Algorithm 4.2. *Let T be the first hitting time from $\vec{x}$ $(\neq \vec{0})$ to $\vec{0}$. A walker starts from $\vec{x}$ and moves right, left, up or down from a current position with equal probability.*

Formula (1) implies $A(z) = f(z)/g(z)$ where

$$f(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} \cos\left(\frac{2\pi k_1 x_1}{m} + \frac{2\pi k_2 x_2}{n}\right)(1 - C_{k_1,k_2}z)^{-1}$$

$$g(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} (1 - C_{k_1,k_2}z)^{-1}.$$

In Algorithms 4.3 and 4.4 we consider a random walk starting from $\vec{x}$ and staying at the current position or moving to one of four neighbors with equal probability. The probability distribution μ is given by $\mu(0,0) = \mu(1,0) = \mu(-1,0) = \mu(0,1) = \mu(0,-1) = \frac{1}{5}$. Then $\hat{\mu}(\chi_{\vec{k}}) = \frac{1}{5} + \frac{4}{5}C_{k_1,k_2}$, $\vec{k} = (k_1, k_2)$.

Algorithm 4.3. *Let T be the first return time to $\vec{0}$. A walker starts from $\vec{0}$ and stays at the current position or moves right, left, up or down.*

Formula (3) implies $A(z) = \frac{1}{5}z + zf(z)/g(z)$ where

$$f(z) = \frac{4}{5} + (1-z)\sum_{\vec{k}\neq\vec{0}} 4C_{k_1,k_2}[5 - z - 4C_{k_1,k_2}z]^{-1}$$

$$g(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} 5[5 - z - 4C_{k_1,k_2}z]^{-1}.$$

Algorithm 4.4. *Let T be the first hitting time from $\vec{x}$ $(\neq \vec{0})$ to $\vec{0}$. A walker starts from $\vec{x}$ and stays at the current position or moves right, left, up, or down.*

Formula (1) implies $A(z) = f(z)/g(z)$ where

$$f(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} 5\cos\left(\frac{2\pi k_1 x_1}{m} + \frac{2\pi k_2 x_2}{n}\right)[5 - z - 4C_{k_1,k_2}z]^{-1}$$

$$g(z) = 1 + (1-z)\sum_{\vec{k}\neq\vec{0}} 5[5 - z - 4C_{k_1,k_2}z]^{-1}.$$

5. Pseudorandom number generators

In this section we present typical pseudorandom number generators. Most of the currently used pseudorandom number generators are based on Linear Congruential Generators, Inversive Congruential Generators, and Lagged Fibonacci Generators [18,27].

Linear Congruential Generators introduced by D.H. Lehmer in 1949 output a sequence of random numbers $\{X_j\}_{j=0}^{\infty}$ according to the rule

$$X_{j+1} = aX_j + b \,(\mathrm{mod}\, M),$$

which is denoted by $LCG(M, a, b)$. They have a lattice structure and the quality of random numbers generated by LCG depends very much on the choices of M, a, b and the seed X_0. The period of $LCG(M, a, b)$ is at most M and the modulus M influences the speed of generation.

Note that $LCG(M, a, b)$ can be viewed as a simple random walk on $(\mathbb{Z}_M, \mu)$ for a probability measure μ with $\mu(1) = \mu(-1) = \frac{1}{2}$ if we fix $a = 1$ and choose b at random from ± 1 with equal probability $\frac{1}{2}$. To increase randomness, several different generators are combined or shuffled. Chung, Diaconis and Graham [5] and Diaconis and Saloff-Coste [10] investigated the properties of the process

$$X_{j+1} = a_j X_j + b_j \,(\mathrm{mod}\, n),$$

where n is odd and a_j, b_j are independent random variables and they might be the output of another pseudorandom number generator or be the output of a truly random source produced by electrical noise or radioactive decay.

In the early sixties, IBM made a pseudorandom number generator, called Randu. It is an LCG(2^{31}, $65539, 0$) and is known as a bad generator. ANSI C and Microsoft C libraries use LCG(2^{31}, $1103515245, 12345$) and LCG($2^{31}, 214013, 2531011$), respectively: but they are not good enough to be used in high precision simulations. Ran0, Ran1, Ran2 and Ran3 were introduced in [31]. Ran0 is an LCG($2^{31} - 1, 16807, 0$) invented by Park and Miller and regarded as a minimal standard generator. Ran1 is a combination of Ran0 and Bay-Durham shuffle. Ran2 is P. L'Ecuyer's algorithm with shuffle, which is a combination of LCG($2147483563, 40014, 0$) and LCG($2147483399, 40692, 0$): it is known to be a very good pseudorandom number generator.

Inversive Congruential Generators invented by Erchenauer and Lehn in 1986 are algorithms of the form

$$X_{j+1} = aX_j^{-1} + b \,(\mathrm{mod}\, M),$$

where M is prime, X^{-1} is an inverse element of X in $\mathbb{Z}_M$ and $0^{-1} = 0$. They are free of the lattice structure.

The Fibonacci Sequence $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$, is defined by the rule $X_{j+2}{=}X_j{+}X_{j+1}$, $j \geqslant 0$. *Lagged Fibonacci Generators* are algorithms of the form

$$X_k = X_{k-j} \pm X_{k-i} \,(\mathrm{mod}\, 2^l), \quad k \geqslant j > i,$$

where l is a positive integer and $X_0, \ldots, X_{j-1}$ are arbitrary integers, which are not all even; choose two constants i and j so that $2^{l-1}(2^j - 1)$ is the period of sequence X_j. Lagged Fibonacci Generators output random numbers very fast because it does not have the multiplication operation.

6. Test methods

In this section we explain how to test pseudorandom number generators using the random walk algorithms suggested in Sections 3 and 4. We consider random variable T which is either the first return time to $\vec{0}$ or the first hitting time from $\vec{x}$ $(\neq \vec{0})$ to $\vec{0}$.

We determine the movements of the random walker using a random sequence $\{X_j\}_{j=0}^{\infty}$ in $\mathbb{Z}_M$ generated by a pseudorandom number generator. For the random walk on $\mathbb{Z}_2^n$, the number of possible movements is n or $n+1$ since each element of $\mathbb{Z}_2^n$ has n neighbors. The possibility of $n+1$ movements is due to the possibility of staying at the current position. Let $\vec{a}=(a_1, a_2, \ldots, a_n)$ be a current position. For $1 \leqslant i \leqslant n$ let $\vec{e}_i = (0, \ldots, 1, \ldots, 0)$ be the binary vector whose components are all zero except that its ith component is equal to 1 and let $\vec{e}_{n+1} = \vec{0}$ for notational convenience. When we have k (equal to n or $n+1$) possible movements, we divide [0,1] into k subintervals $I_i = [\frac{i-1}{k}, \frac{i}{k})$, $1 \leqslant i \leqslant k$, and check which subinterval the real number X_j/M falls into: if $X_j/M \in I_i$, then the next movement is $\vec{a} + \vec{e}_i$. For the random walk on $\mathbb{Z}_m \times \mathbb{Z}_n$, the number of possible movements is four or five. Let $\vec{a}=(a_1, a_2)$ be a current position. When we have k (equal to 4 or 5) possible movements, we divide [0,1] into k subintervals $I_i = [\frac{i-1}{k}, \frac{i}{k})$, $1 \leqslant i \leqslant k$, and check where X_j/M falls into: if $X_j/M \in I_i$, then a walker moves from $\vec{a}$ to $\vec{a} + \vec{r}_i$ where $\vec{r}_1 = (1, 0)$, $\vec{r}_2 = (0, 1)$, $\vec{r}_3 = (-1, 0)$, $\vec{r}_4 = (0, -1)$ and $\vec{r}_5 = (0, 0)$. We continue this procedure until the walker arrives at the final point $\vec{0}$ and measure the required number of steps, which correspond to the first return time or the first hitting time. We repeat this N times.

Good pseudorandom number generators make the movements fairly random and the random variable $Z = \frac{\bar{\mu} - \mathbb{E}(T)}{\sigma(T)/\sqrt{N}}$ approximates the standard normal distribution for sufficiently large sample size N, where $\bar{\mu}$ is a sample mean. In the tests we expect $|Z| < 1.96$ with probability 0.95 and $|Z| < 2.58$ with probability 0.99.

7. Experimental results and conclusion

We test Randu, namely, a Linear Congruential Generator of the form

$$X_{j+1} = 65539 X_j \,(\mathrm{mod}\, 2^{31})$$

and Lagged Fibonacci Generators of the forms

$$\mathrm{LFG1}: X_j = X_{j-55} - X_{j-24} \,(\mathrm{mod}\, 2^{30})$$

$$\mathrm{LFG2}: X_j = X_{j-127} - X_{j-30} \,(\mathrm{mod}\, 2^{30})$$

$$\mathrm{LFG3}: X_j = X_{j-100} - X_{j-37} \,(\mathrm{mod}\, 2^{30})$$

using random walk algorithms on $\mathbb{Z}_2^n$, $8 \leqslant n \leqslant 17$, $\mathbb{Z}_{50\times 100}$ and $\mathbb{Z}_{100\times 100}$. When we study the first return time, we fix the starting point $\vec{0}$. When we study the first hitting time, we fix the final point $\vec{0}$ and take the starting point $\vec{1}$ for $\mathbb{Z}_2^n$; $(25, 50)$ for $\mathbb{Z}_{50\times 100}$; $(50, 50)$ for $\mathbb{Z}_{100\times 100}$.

The probability generating functions $A(z) = \sum_{k=0}^{\infty} \Pr(T = k) z^k$ of the first hitting time (or the first return time) T, derived in Sections 3 and 4, provide the theoretical values of expectation $\mathbb{E}(T)$ and variation $\sigma^2(T)$ by $\mathbb{E}(T) = A'(1)$ and $\sigma^2(T) = A''(1) + A'(1) - A'(1)^2$. They are presented in Fig. 1.

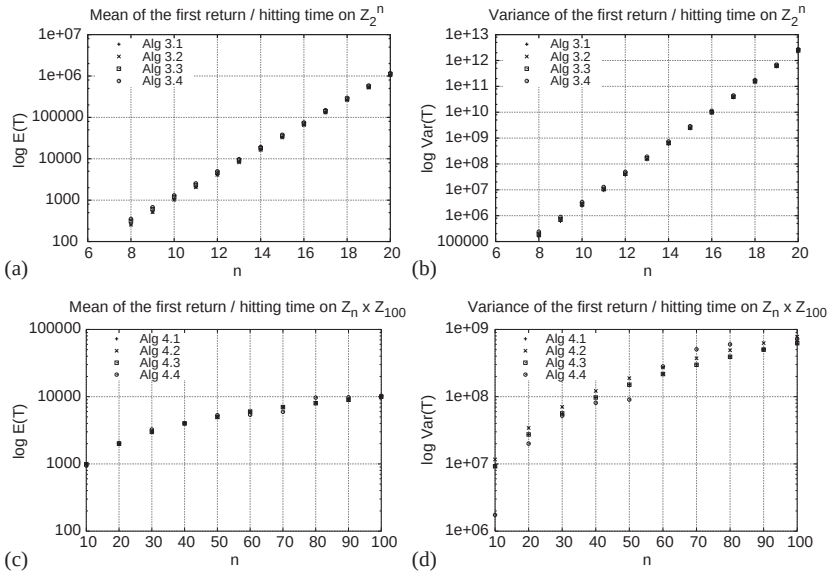

Fig. 1. (a) Mean of the first return time/hitting time of Algorithms 3.1–3.4; (b) variance of algorithms 3.1–3.4; (c) mean of algorithms 4.1–4.4; (d) variance of algorithms 4.1–4.4.

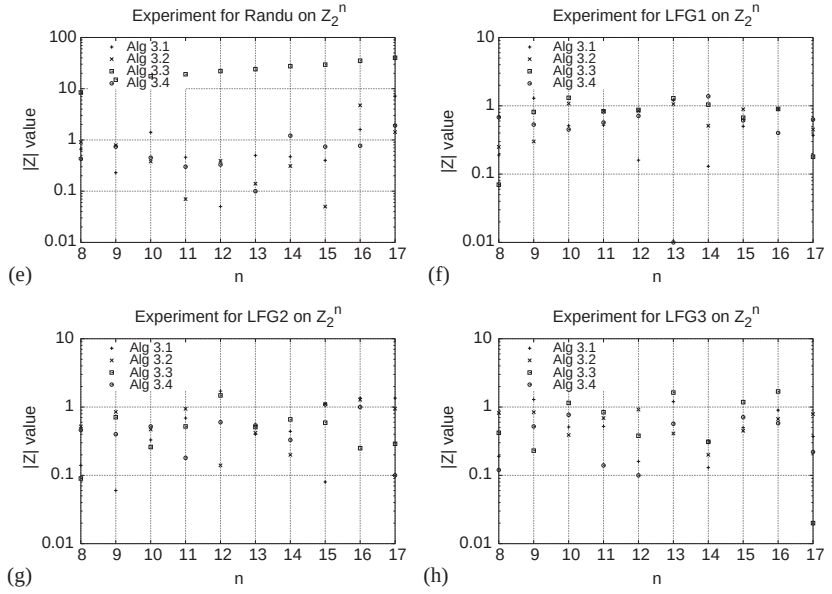

Fig. 2. $|Z|$ values of (e) Randu, (f) LFG1, (g) LFG2 and (h) LFG3 using Algorithms 3.1–3.4.

Fig. 2 and Table 1 present the $|Z| = \left| \frac{\bar{\mu} - \mathbb{E}(T)}{\sigma(T)/\sqrt{N}} \right|$ values of experiments for Randu, LFG1, LFG2 and LFG3 when we take $N = 10^5$: our implementation result says that Randu fails the test, but LFG1, LFG2 and LFG3 succeed it, which implies that Randu is a bad generator, but LFG1, LFG2 and

Table 1
$|Z|$ values of Randu, LFG1, LFG2 and LFG3 using Algorithms 4.1–4.4

	Algorithm 4.1	Algorithm 4.2	Algorithm 4.3	Algorithm 4.4
$\mathbb{Z}_{50} \times \mathbb{Z}_{100}$				
Randu	0.88	92.53	1.03	111.84
LFG1	0.15	1.07	0.06	0.35
LFG2	0.54	0.98	1.48	1.23
LFG3	0.65	0.94	0.19	0.73
$\mathbb{Z}_{100} \times \mathbb{Z}_{100}$				
Randu	1.18	106.44	0.50	107.06
LFG1	0.64	0.99	1.19	0.31
LFG2	0.17	0.55	0.42	0.39
LFG3	0.42	1.27	0.30	0.02

LFG3 are good generators. Moreover it indicates that the suggested random walk algorithms are useful to test the efficiency of other pseudorandom number generators.

Acknowledgements

The author would like to thank Chihurn Kim for helpful advice on computer simulations.

References

[1] D. Aldous, P. Diaconis, Shuffling cards and stopping times, Amer. Math. Monthly 93 (5) (1986) 333–348.
[2] D. Aldous, J. Fill, Reversible Markov Chains and Random Walks on Graphs, in preparation.
[3] B. Bollobás, Modern Graph Theory, Springer, New York, 1998.
[4] R. Bubley, M. Dyer, Path coupling: a technique for proving rapidly mixing in Markov chains, Proceedings of the 38th IEEE Symposium on Foundations of Computer Science, IEEE Computer Science Press, Silver Spring, MD, 1997.
[5] F. Chung, P. Diaconis, R. Graham, Random walks arising in random number generation, Ann. Probab. 15 (1987) 1148–1165.
[6] D. Copersmith, P. Doyle, P. Raghavan, M. Snir, Random walks on weighted graphs and applications to on-line algorithms, J. Amer. Math. Soc. 40 (1993) 421–453.
[7] P. Diaconis, Group Representations in Probability and Statistics, Inst. Math. Stat., Hayward, 1988.
[8] P. Diaconis, The cutoff phenomenon in finite Markov chains, Proc. Nat. Acad. Sci. USA 93 (4) (1996) 1659–1664.
[9] P. Diaconis, L. Saloff-Coste, Comparison theorems for reversible Markov chains, Ann. Appl. Probab. 3 (1993) 696–730.
[10] P. Diaconis, L. Saloff-Coste, Moderate growth and random walk on finite groups, Geom. Funct. Anal. 4 (1) (1994) 1–36.
[11] P. Diaconis, D. Stroock, Geometric bounds for eigenvalues of Markov chains, Ann. Appl. Probab. 1 (1991) 36–61.
[12] P. Doyle, J. Snell, Random Walks and Electric Networks, The Carus Mathematical Monographs 22, Mathematical Association of America, 1984.
[13] R. Durrett, Probability: Theory and Examples, 2nd Edition, Duxbury Press, Belmont, 1996.
[14] M. Dyer, A. Frieze, R. Kannan, A random polynomial time algorithm for approximating the volume of convex bodies, J. of ACM 38 (1991) 1–17.

[15] J. Fill, An interruptible algorithm for perfect sampling via Markov chains, The Ann. Appl. Probab. 8 (1) (1998) 131–162.
[16] J. Fill, M. Machida, D. Murdoch, J. Rosenthal, Extension of Fill's perfect rejection sampling algorithm to general chains, Random Struct. Algorithms 17 (2000) 290–316.
[17] L. Flatto, A. Odlyzko, D. Wales, Random shuffles and group representations, Ann. Probab. 13 (1985) 154–178.
[18] J. Gentle, Random number generation and Monte Carlo methods, Statistics and Computing, Springer, Berlin, 1998.
[19] D. Gluck, First hitting times for some random walks on finite groups, J. Theoret. Probab. 12 (3) (1999) 739–755.
[20] S. Janson, D. Randall, J. Spencer, Random dyadic tilings of the unit square, Random Struct. and Algorithms 21 (2002) 225–251.
[21] M. Jerrum, Mathematical foundations of the Markov chain Monte Carlo method, in: M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, B. Reed (Eds.), Probabilistic Methods for Algorithmic Discrete Mathematics, Algorithms and Combinatorics, Vol. 16, Springer, Berlin, 1998.
[22] M. Jerrum, A. Sinclair, The Markov chain Monte Carlo method: an approach to approximate counting and integration, in: D. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing, Boston, 1996.
[23] M. Kac, On the notion of recurrence in discrete stochastic processes, Bull. Amer. Math. Soc. 53 (1947) 1002–1010.
[24] M. Kang, First hitting times of simple random walks on graphs with congestion points, Internat. J. Math. Math. Sci. 2003 (30) (2003) 1911–1922.
[25] M. Kang, Random walks on a finite graph with congestion points, Appl. Math. Comput., to appear.
[26] A. Kholodenko, Some thoughts about random walks on figure eight, J. Physica A 289 (3–4) (2001) 377–408.
[27] D. Knuth, The Art of Computer Programming, Vol. 2, 3rd Edition, Addison-Wesley, Reading, MA, 1997.
[28] M. Luby, D. Randall, A. Sinclair, Markov chain algorithms for planar lattice structures, SIAM J. Comput. 31 (2001) 167–192.
[29] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, New York, 1995.
[30] I. Peterson, The Jungles of Randomness, Wiley, New York, 1998.
[31] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, Numerical Recipes in C, 2nd Edition, Cambridge University Press, Cambridge, 1992.
[32] J. Propp, D. Wilson, Exact sampling with coupled Markov chains and applications to statistical mechanics, Random Struct. Algorithms 9 (1996) 223–252.
[33] P. Tetali, Random walks and the effective resistance of networks, J. Theoret. Probab. 4 (1991) 101–109.
[34] P. Tetali, Design of on-line algorithms using hitting times, SIAM J. Comput. 28 (4) (1999) 1232–1246.
[35] E. Vigoda, Improved bounds for sampling colorings, J. Math. Phys. 41 (3) (2000) 1555–1569.
[36] D. Wilson, How to couple from the past using a read-once source of randomness, Random Struct. Algorithms 16 (2000) 85–113.