Randomness in algorithm design

Shuji Kijima
Kyushu University, Fukuoka, 819-0395, Japan
E-mail: kijima@inf.kyushu-u.ac.jp

Abstract. What is the property that a randomized algorithm really requires for randomness? This article exhibits some topics related to randomization and derandomization in computation, the topics which the author has been concerned with motivated by the above question.

1. Introduction
Randomness is a powerful and indispensable tool for computation nowadays, supported by the probability theory standing on a long history. “What is the property that a randomized algorithm really requires for randomness?” Motivated by the question, the author has been investigating randomization and derandomization of algorithms. This article exhibits some topics related to this subject.

To demonstrate the power of randomness, we are concerned with a fundamental problem of finding frequent items in a data stream in Section 2. For the problem, we have developed a randomized approximation algorithm using memory of only \(O(\log \log n)\) bits in [49]. Our algorithm is based on an extension of probabilistic counting using memory of only \(O(\log \log n)\) bits, while it is known that any deterministic algorithm for counting requires \(\Omega(\log n)\) bits, even for approximation. In Section 3, we briefly review the Markov chain Monte Carlo (MCMC) method, which is a sophisticated and useful technique for random sampling. The coupling from the past, devised by Propp and Wilson [50], is a wonderful algorithm for MCMC, which provides perfect sampling, meaning that a random sample exactly according to the limit distribution, by an ingenious simulation of Markov chain. We briefly review the coupling from the past algorithm in Section 4. In Section 5, we introduce the topic of deterministic random walks on finite graphs, which is an approach to replacing a random walk by a deterministic process, motivated by a question if it is possible to derandomize an MCMC.

2. Finding frequent items in a stream
2.1. Problem
Let us consider a problem of finding frequent items in a data stream (cf. [49]). Let \(\Sigma\) be a finite set\(^1\), which we call “items”, and let \(x = (x_1, x_2, \ldots) \in \Sigma^*\) denote a data stream of items. For instance, \(\Sigma\) denotes a set of IP addresses and \(x\) denotes an access-log of web page in a day. For another example, \(\Sigma\) denotes a set of goods in a supermarket and \(x\) denotes a POS data in a day. Suppose a threshold \(\theta \in (0, 1)\) is prescribed, and we want to know frequent items \(s \in \Sigma\),

\(^1\) For simplicity, we in this paper assume that \(\Sigma\) is finite where every item is identified with a constant number \(\sigma := \lceil \log((|\Sigma| + 1) \rceil\) of bits.
each of which appears at least $\theta \cdot n$ times in an input data stream $x = (x_1, x_2, \ldots, x_n)$. How large memory space is required to find frequent items in $x$, without knowing the value of $n$ in advance?

Boyer and Moore [3, 4] gave an algorithm, known as majority algorithm, which solves the problem for $\theta = 1/2$ using $O(\log n)$ bits, while Fischer and Salzburg [20] independently gave essentially the same algorithm. Karp, Shenker, and Papadimitriou [32] showed a lower bound that any online algorithm requires $\Omega(|\Sigma| \log(n/|\Sigma|))$ bits to find the frequent items exactly. Karp, Shenker, and Papadimitriou [32] also gave a simple and elegant deterministic approximation algorithm, which allows false positive outputs using memory of $O(\theta^{-1} \log n)$ bits. Demaine, López-Ortiz, and Munro [9] gave an algorithm essentially the same as [32]. Manku and Motwani [44] proposed another deterministic algorithm, called lossy counting, which is based on approximate counting using $O(\varepsilon^{-1} \log(\varepsilon^{-1})n)$ bits, that outputs items of frequency more than $(\theta - \varepsilon)$ for $\varepsilon \in (0, 1)$.

Many randomized algorithms are also known for the problem. Toivonen [54] proposed an algorithm using memory of $O(\varepsilon^{-2} \log \varepsilon^{-1} \log n)$ bits, which is based on uniform sampling of data in a stream, and outputs all frequent items with probability $1 - \delta$. Manku and Motwani [44] gave another algorithm, called sticky sampling, which uses memory of $O(\varepsilon^{-1} \log \varepsilon^{-1} \log n)$ bits and outputs all frequent items with probability $1 - \delta$. Their algorithm samples items by adaptively changing the sampling rate according to the number of items that have arrived.

Each of deterministic or randomized approximation algorithms above mentioned uses memory of $O(\log n)$ bits in terms of $n$, respectively. Thus, it can be a natural question: Is there an algorithm which uses memory of $o(\log n)$ for the problem without knowing $n$ in advance?

2.2. Probabilistic counting

Let us consider a simpler problem — just count the length $n$ of an input stream $x = (x_1, \ldots, x_n) \in \Sigma^n$ where $\Sigma$ denotes the set of items. A trivial algorithm uses memory of $\lceil \log(n+1) \rceil$ bits for counting exact $n$, and this is best possible because it demands $\lceil \log(n+1) \rceil$ bits to represent exact $n$. It is thus natural to ask if it is possible to approximate $n$, e.g., to compute $\log n$, using memory of $o(\log n)$ bits. Unfortunately, any deterministic algorithm requires a working space of $\log n$ bits even for an approximation of $\log n$.

Morris [47] gave a fascinating randomized approximation algorithm, called probabilistic counting, for computing $\log n$. The algorithm is as follows; set the initial value $h = 0$, and increment $h$ by one with probability $2^{-h}$ for every input item. Then, the expectation of $h$ is shown to be $\log(n+2)$ (see also [21]). Morris also generalized it to treat with a sampling rate $a^{-h}$ for any constants $a > 0$. Flajolet [21] gave detailed analyses on the variance of the probabilistic counting.

We can see that the probabilistic counting uses memory of only $O(\log \log n)$ bits, which implies that a randomized algorithm reduces the space complexity of the problem, compared with a deterministic algorithm.

2.3. Finding frequent items in a stream using $O(\log \log n)$ bits

Now, back to the problem of finding frequent items in a stream. Let $\Sigma$ denote a finite set of items, and assume that every item is identified with a constant $\gamma := \lceil \log |\Sigma| \rceil$ bits. For a sequence $x = (x_1, \ldots, x_n)$ of items in $\Sigma$, let $f(s) \in \{0, \ldots, n\}$ for $s \in \Sigma$ denote the frequency of $s$ in $x$. Given a (constant) $\theta \in (0, 1)$, we say that an item $s \in \Sigma$ is frequent (in $x$) if $f(s) \geq \theta n$.

In our approximation algorithm, suppose that parameters $\theta \in (0, 1)$, $\gamma \in (0, 1)$ and $\delta \in (0, 1)$ are prescribed by a user. Then with probability at least $1 - \delta$, the output of our algorithm contains all frequent items in $\{s \in \Sigma \mid f(s) \geq \theta n\}$ and does not contain any rare item in $\{s \in \Sigma \mid f(s) < (1 - \gamma)\theta n\}$. The following shows our algorithm for an input stream $x = (x_1, \ldots, x_n)$, in which $b \in \mathbb{N}$ is a parameter for approximation. Let $K$ denote a data
structure which stores $2^b$ pairs of an item and a number $(s, K[s])$, where $s$ uses $\sigma$ bits, and $K[s]$ uses $b$ bits.

**Algorithm 2.1.** (\cite{49})

0. **Initialize** $K$. **Set** “exponent” $h := 0$.
1. **Read** an input $x_i$ if it exists, otherwise **goto** 5. Suppose $x_i = s^*$ for convenience.
2. **“Record”** $s^*$ in $K$ with probability $2^{-h}$:
   - add $s^*$ in $K$ unless $s^*$ in $K$, and increment $K[s^*]$ by one.
3. **Unless** $\sum_{s' \in K} K[s'] = 2^b$, **goto** 1.
4. **“Flush”**:
   - 4-(i) **Increment** $h$ by one.
   - 4-(ii) **For** each $s$ in $K$,
     - choose $k$ with probability $(K[s]) / 2^h$, and
     - set $K[s] := k$.
   - 4-(iii) **Delete** symbol $s \in K$ if $K[s] = 0$.
5. **Output** every $s$ satisfying that $K[s] \geq (1 - \gamma/2)\theta \sum_{s' \in K} K[s']$.

Let $h^*$ denote the value of exponent when Algorithm 2.1 terminates. Then, we can show that Algorithm 2.1 uses $(b + \sigma)2^b + [\lg h^*]$ bits, and an extra working space of $O(\lg h^*)$ bits for Steps 2 and 4. We also remark that the number of items output by Algorithm 2.1 is at most $1/(\theta(1-\gamma/2))$.

**Theorem 2.1.** (\cite{49}) Arbitrarily given $\theta \in (0, 1)$, $\gamma \in (0, 1)$ and $\delta \in (0, 1)$, set

$$b \geq \lceil \lg ((\theta\gamma/2)^{-2}\ln(3((1-\gamma/2)\theta\delta)^{-1})) \rceil + 3.$$ 

Then the output of Algorithm 2.1 contains all frequent items in \{ $s \in \Sigma$ | $f(s) \geq \theta n$ \} and does not contain any rare item in \{ $s \in \Sigma$ | $f(s) < (1 - \gamma)\theta n$ \}, with probability at least $1 - \delta$, for any input stream $x = (x_1, \ldots, x_n)$.

Theorem 2.1 indicates that our algorithm uses memory of

$$2^b(b + \sigma) + \lg \lg n \simeq 32 \ln \left( \frac{3}{(1-\gamma/2)\theta\delta} \right) \ln \left( \frac{\ln \left( \frac{3}{(1-\gamma/2)\theta\delta} \right)}{(\gamma\theta)^2} \right) + \lg \lg n = O \left( \frac{\log^2(\theta^{-1})}{\theta^2} + \log \log n \right)$$

bits.

3. **Markov chain Monte Carlo**—for sampling combinatorial objects

Random sampling plays a fundamental role in randomized algorithms, especially from the viewpoint of computational efficiency. It is known, in the context of polynomial time computability, that there is a strong connection between sampling and approximate counting of combinatorial objects through the self-reducibility of the objects. For example, Dyer, Frieze, Kannan \cite{19} gave a fully polynomial time randomized approximation scheme (FPRAS) for the volume computation of a convex body in a high dimensional space, which is a $\#P$-hard problem, using random sampling in a convex body and self reducibility. As another example, Jerrum, Sinclair, Vigoda \cite{30} gave an FPRAS for the computation of the permanent of a $n \times n$ non-negative matrix, which is also $\#P$-hard. Both algorithms are based on the Markov chain Monte Carlo method.
The Markov chain Monte Carlo is a Monte Carlo method using a Markov chain for random sampling. The idea is simple: (1) design a Markov chain whose limit distribution is desired distribution, and (2) sample from the limit distribution simulating the chain. This section briefly reviews the MCMC method.

3.1. Designing a Markov chain with a desired distribution

To realize (1), there is a framework for designing a Markov chain with a desired limit distribution using reversible Markov chain, such as cerebrated Metropolis-Hastings, or Gibbs sampler.

Here, we are concerned with a finite Markov chain $\mathcal{M}$, defined by a transition matrix $P$ on finite states $\Omega$. A Markov chain is irreducible if $\exists t > 0$, $\Pr(X^t = y \mid X^0 = x) > 0$ holds for any pair $\{x, y\} \in \binom{\Omega}{2}$, where $X^t \in \Omega$ denotes the state of the Markov chain at time $t$. A Markov chain is aperiodic if $\forall x \in \Omega$, $\gcd(\{t \in \mathbb{Z}_{\geq 0} \mid \Pr(X^t = x \mid X^0 = x) > 0\}) = 1$ holds, where $\gcd(A)$ denotes the greatest common deviser of $A \subseteq \mathbb{Z}_{\geq 0}$. A Markov chain is ergodic if it is irreducible and aperiodic. It is well-known that an ergodic Markov chain has a unique stationary distribution $\pi$, which is a distribution satisfying $\pi P = \pi$, and the limit distribution is $\pi$.

The following Theorem provides a Markov chain with a desired stationary distribution.

**Theorem 3.1.** (cf. [26, 43]) Suppose $\mathcal{M}$ is an ergodic Markov chain defined by a transition matrix $P$ on a state space $\Omega$. If a function $f : \Omega \to \mathbb{R}_{> 0}$ satisfies the following detailed balance equation

$$f(x)P(x, y) = f(y)P(y, x), \quad \forall\{x, y\} \in \binom{\Omega}{2}$$

then the stationary distribution $\pi$ of $\mathcal{M}$ satisfies $\pi(x) = C \cdot f(x)$ for any $x \in \Omega$ with a constant $C$, where $P(x, y)$ denotes the transition probability from $x$ to $y$.

A Markov chain satisfying the detailed balance equation is said reversible. Many Markov chains appearing in MCMC are reversible.

3.2. Mixing time

On the other hand, estimation the speed of convergence into limit distribution, called mixing time of the Markov chain, for the purpose of (2) is not easy in general. Remark that “convergence” does not mean the convergence into a state of the Markov chain, and hence it is also difficult in practice to determine if the Markov chain converges to its limit distribution.

Given a pair of probability distributions $\nu_1$ and $\nu_2$ on the finite state space $\Omega$, the total variation distance between $\nu_1$ and $\nu_2$ is defined by

$$d_{TV}(\nu_1, \nu_2) \overset{\text{def.}}{=} \frac{1}{2} \sum_{x \in \Omega} |\nu_1(x) - \nu_2(x)|.$$ 

For any positive $\varepsilon < 1$, the mixing time $\tau(\varepsilon)$ of $\mathcal{M}$ with a state space $\Omega$ is defined by

$$\tau(\varepsilon) \overset{\text{def.}}{=} \max_{x \in \Omega} \min\{t \mid \forall s \geq t, \ d_{TV}(\pi, P^s_x) \leq \varepsilon\}$$

where $P^s_x$ denotes the distribution at time $s$ of $\mathcal{M}$ starting from $x \in \Omega$. The definition of the mixing time implies that the total variation distance between the stationary distribution and the distribution of $\mathcal{M}$ after $\tau(\varepsilon)$ transitions is at most $\varepsilon$. We say a Markov chain is rapidly mixing if $\tau(\varepsilon) \leq \text{poly}(\text{input size}, \varepsilon^{-1})$.

Mixing time is deeply related to the eigenvalue of the transition matrix. For instance, the following theorem is well known (see e.g., [51, 43]).
Theorem 3.2. Suppose a Markov chain $M$ defined by a transition matrix $P$ is reversible and ergodic. Let $\lambda_i$ ($i \in \{1, \ldots, |\Omega|\}$) denote the (multiple) eigenvalues of $P$, where without loss of generality we may assume that $\lambda_1 = 1$. Then, for any $\varepsilon$ ($0 < \varepsilon < 1$),

$$\frac{\lambda^* \ln((2\varepsilon)^{-1})}{1 - \lambda^*} \leq \tau(\varepsilon) \leq \frac{\ln(\pi_{\min}^{-1}) + \ln(\varepsilon^{-1})}{1 - \lambda^*}$$

holds where $\lambda^* \overset{\text{def.}}{=} \max\{\lambda_i | i \in \{2, \ldots, |\Omega|\}\}$ and $\pi_{\min} \overset{\text{def.}}{=} \min\{\pi(x) | x \in \Omega\}$ for the stationary distribution $\pi$ of $M$. Note that $\lambda^* < 1$ holds.

4. Perfect sampling

In stead of estimating the mixing time of a Markov chain, Propp and Wilson [50] devised an algorithm which enables perfect sampling, meaning that the algorithm provides a random sample according to exactly limit distribution. The algorithm is an ingenious simulation of a Markov chain, called coupling from the past (CFTP). Here we briefly review the CFTP.

4.1. Coupling from the past

Suppose that we have an ergodic Markov chain $MC \leftarrow X'$ with finite state space $\Omega$ and transition matrix $P$. The transition rule of the Markov chain $X \rightarrow X'$ can be described by a deterministic function $\phi$: $\Omega \times [0, 1) \rightarrow \Omega$, called an update function, as follows. Given a random number $\lambda$ uniformly distributed over $[0, 1)$, an update function $\phi$ satisfies $Pr(\phi(x, \lambda) = y) = P(x, y)$ for any $x, y \in \Omega$. We can realize the Markov chain by setting $X' = \phi(X, \lambda)$. Clearly, an update function corresponding to the given transition matrix $P$ is not unique. The result of transitions of the chain from the time $t_1$ to $t_2$ ($t_1 < t_2$) with a sequence of random numbers $\lambda = (\lambda[t_1], \lambda[t_1+1], \ldots, \lambda[t_2-1]) \in [0, 1)^{t_2-t_1}$ is denoted by $\Phi_{t_1}^{t_2}(x, \lambda) : \Omega \times [0, 1)^{t_2-t_1} \rightarrow \Omega$ where

$$\Phi_{t_1}^{t_2}(x, \lambda) \overset{\text{def.}}{=} \phi(\phi(\cdots(\phi(x, \lambda[t_1]), \ldots, \lambda[t_2-2]), \lambda[t_2-1])).$$

We say that a sequence $\lambda \in [0, 1)^{[T]}$ satisfies the coalescence condition, when $\exists y \in \Omega$, $\forall x \in \Omega$, $y = \Phi_0^T(x, \lambda)$.

With these preparations, standard Coupling From The Past algorithm is expressed as follows.

Algorithm 4.1. (Coupling from the past [50])

Step 1. Set the starting time period $T := -1$ to go back, and let $\lambda$ be an empty sequence.

Step 2. Generate random real numbers $\lambda[T], \lambda[T+1], \ldots, \lambda[(T/2)-1] \in [0, 1)$, and insert them to the head of $\lambda$ in order, i.e., put $\lambda := (\lambda[T], \lambda[T+1], \ldots, \lambda[-1])$.

Step 3. Start a chain from each element $x \in \Omega$ at time period $T$, and run each chain to time period 0 according to the update function $\phi$ with the sequence of numbers in $\lambda$. (Here we note that every chain uses the common sequence $\lambda$.)

Step 4. [Coalescence check] The state obtained at time period 0 can be denoted by $\Phi^0_T(x, \lambda)$.

(i) If $\exists y \in \Omega$, $\forall x \in \Omega$, $y = \Phi^0_T(x, \lambda)$, then return $y$ and stop.

(ii) Else, update the starting time period $T := 2T$, and go to Step 2.

Theorem 4.1. ([50]) Let $MC$ be an ergodic finite Markov chain with state space $\Omega$, defined by an update function $\phi: \Omega \times [0, 1) \rightarrow \Omega$. If the CFTP algorithm (Algorithm 4.1) terminates with probability 1, then the obtained value is a realization of a random variable exactly distributed according to the stationary distribution.

It is also well known that the time to couple is deeply related to the mixing time (see e.g., [50, 35]). We define the coalescence time $T^*$ by

$$T^* \overset{\text{def.}}{=} \min\{t < 0 | \exists y \in \Omega, \forall x \in \Omega, y = \Phi^0_T(x, \lambda)\},$$

which denotes the minimum number of steps to obtain a sample by Algorithm 4.1.
Let \( N \) be a deterministic random walk on a graph. Instead of distributing tokens to randomly chosen neighbors, the rotor-router model replaces the random walk with a deterministic process analogous to the Propp machine. Propp machine deterministically serves the neighbors in a fixed order by associating to each edge \( e \) a deterministic update function \( \phi \) is called monotone (with respect to \( \geq \)) if \( \forall \lambda \in [0, 1), \forall x, y \in \Omega, x \geq y \Rightarrow \phi(x, \lambda) \geq \phi(y, \lambda) \). We also say that a chain is monotone if the chain has a monotone update function.

**Theorem 4.4.** Suppose that a Markov chain defined by an update function \( \phi \) is monotone with respect to a partially ordered set of states \( (\Omega, \succeq) \), and \( \exists x_{\text{max}}, \exists x_{\text{min}} \in \Omega, \forall x \in \Omega, x_{\text{max}} \succeq x \succeq x_{\text{min}} \). Then the CFTP algorithm (Algorithm 4.1) terminates with probability 1, and a sequence \( \lambda \in [0, 1)^{|T|} \) satisfies the coalescence condition, i.e., \( \exists y \in \Omega, \forall x \in \Omega, y = P_T^0(x, \lambda) \), if and only if \( P_T^0(x_{\text{max}}, \lambda) = P_T^0(x_{\text{min}}, \lambda) \).

When the given Markov chain satisfies the conditions of Theorem 4.3, we can modify Algorithm 4.1 by substituting Step 4 (a) by

**Step 4.** (a)' If \( \exists y \in \Omega, y = P_T^0(x_{\text{max}}, \lambda) = P_T^0(x_{\text{min}}, \lambda) \), then return \( y \) and stop.

The algorithm obtained by the above modification is called a monotone CFTP algorithm. Propp and Wilson [50] gave a monotone CFTP for Ising model. Kijima and Matsui [35] gave a monotone Markov chain for sampling two rowed contingency tables (see also [56]).

## 5. Deterministic random walk

The rotor-router model, also known as the Propp machine [8], is a deterministic process analogous to a random walk on a graph. Instead of distributing tokens to randomly chosen neighbors, the Propp machine deterministically serves the neighbors in a fixed order by associating to each vertex a “rotor-router” pointing to one of its neighbors. The Propp machine sometimes appears under the name of deterministic random walk, meaning a “derandomized, hence deterministic, version of a random walk.”

### 5.1. Rotor-router model

Let \( G = (V, \mathcal{E}) \) be a multidigraph (i.e., directed graph with multiple edges) with vertex set \( V \) and edge set \( \mathcal{E} \), where \( n = |V| \) and \( m = |\mathcal{E}| \). Here, we assume that \( G \) is strongly connected [8, 34, 31]. Let \( N(v) \) denote the multiset of vertices consisting of destinations of outgoing edges from \( v \in V \) in \( G \), and let \( \delta(v) = |N(v)| \), i.e., the outdegree of \( v \in V \) in \( G \). Note that \( v \) is a member of \( N(v) \) if \( \mathcal{E} \) includes a self-loop at \( v \).

Let \( P \) be an \( n \times n \) matrix representing a standard random walk on \( G \), meaning that

\[
P(u, v) = \frac{|\{a \in \mathcal{E} | a = (u, v)\}|}{\delta(u)}.
\]

Since \( G \) is strongly connected, \( P \) is irreducible. In this paper, we consider random walks of \( M \) tokens on \( G \), in which each token independently moves according to the transition matrix \( P \). Let \( \mu^{(0)} = (\mu_{v_1}^{(0)}, \ldots, \mu_{v_n}^{(0)}) \in \mathbb{Z}_{\geq 0}^n \) denote an initial configuration of tokens, and let \( \mu^{(t)} = (\mu_{v_1}^{(t)}, \ldots, \mu_{v_n}^{(t)}) \in \mathbb{R}_{\geq 0}^n \) denote the expected configuration of tokens according to \( P \) at time \( t \in \mathbb{Z}_{\geq 0} \), i.e., \( \mu^{(t)} = \mu^{(0)}P^t \).
The rotor-router model is a deterministic process analogous to $\mu^{(t)}$. Let $\chi_v^{(0)}(=\mu^{(0)})$ be an initial configuration of tokens, and let $\chi_v^{(t)}$ denote a configuration of tokens in the rotor-router model at time $t$. In the rotor-router model, an ordering $\rho_v(0), \ldots, \rho_v(\delta(v) - 1)$ of $N(v)$ is prescribed on each vertex $v \in V$. We define
\[
\rho_v(i) \overset{\text{def}}{=} \rho_v(i \mod \delta(v)) \quad \left(= \rho_v \left( i - \delta(v), \left\lfloor \frac{i}{\delta(v)} \right\rfloor \right) \right)
\]
for all $i \in \mathbb{Z}_{\geq 0}$ by extension. This ordering $\rho_v$ is called a rotor-router on $v$, which sequentially launches all $\chi_v^{(t)}$ tokens at each time $t \in \mathbb{Z}_{\geq 0}$. More precisely, since $\chi_v^{(0)}$ tokens are located on $v \in V$ at time $t = 0$, during the time interval $[0, 1)$, the rotor-router on $v$ launches each token $j$ ($j \in \{1, \ldots, \chi_v^{(0)}\}$) into vertex $\rho_v(j - 1) \in N(v)$, where they arrive at the neighbors of $v$ at time $t = 1$. Since at time $t \geq 1$, $\chi_v^{(t)}$ tokens are located on $v \in V$, and the rotor-router on $v$ points to $\rho_v(\sum_{s=0}^{t-1} \chi_v^{(s)})$, the rotor-router on $v$ launches each token $j$ ($j \in \{1, \ldots, \chi_v^{(t)}\}$) into vertex $\rho_v(j - 1 + \sum_{s=0}^{t-1} \chi_v^{(s)}) \in N(v)$ during the time interval $[t, t+1)$, where they arrive at the neighbors of $v$ at time $t+1$.

5.2. Discrepancy

Our goal is to estimate the discrepancy $\left| \chi_w^{(T)} - \mu_w^{(T)} \right|$ for $w \in V$ and $T \geq 0$. We have obtained the following results in [34, 31].

**Theorem 5.1.** ([34]) If all the eigenvalues of $P$ are nonnegative, then for any initial configuration of tokens and routers, we have
\[
\left| \chi_w^{(T)} - \mu_w^{(T)} \right| \leq (2m - n) \max_{i \in \{1, \ldots, \kappa\}} n_i + n + 3 \leq 4mn + O(m)
\]
for any $w \in V$ and $T \geq 0$, where $n_i (i \in \{1, \ldots, \kappa\})$ denotes the size of the $i$-th Jordan block of the Jordan normal form of $P$ with $\kappa$ the number of Jordan blocks.

**Theorem 5.2.** ([34]) There exist a multidigraph and an initial configuration of tokens and routers such that $\left| \chi_w^{(T)} - \mu_w^{(T)} \right| = \Omega(m)$ holds for any node $w$ and any positive integer $T$, even when the corresponding $P$ is positive semidefinite, hence all the eigenvalues of $P$ are nonnegative and $P$ is diagonalizable.

**Theorem 5.3.** ([31]) Let $P$ be the transition matrix of the standard random walk on a strongly connected multidigraph $G$, and let $J = B^{-1}PB$ be a Jordan normal form of $P$ where each column vector of $B$ is normalized to norm one. Let $\lambda^* \overset{\text{def}}{=} \max\{ |\lambda_i| \mid \lambda_i$ is an eigenvalue of $P$, $|\lambda_i| \neq 1 \}$, and let $\alpha^*$ be the maximum in the absolute values of elements of $B^{-1}$. Then, for any initial configuration of tokens and routers,
\[
\left| \chi_w^{(T)} - \mu_w^{(T)} \right| \leq \frac{2n(n - r)(m - n)\alpha^*}{1 - \lambda^*}
\]
holds for any $w \in V$ and $T \geq 0$, where $r$ is the periodicity of $P$.

5.3. Deterministic random walk for irrational transition probabilities

Recently Shiraga et al. [52] gave a functional router model, which simulates Markov chains with irrational transition probabilities, in a similar fashion to the rotor-router model. They also gave a bound of the discrepancy for the functional router model as follows.
Theorem 5.4. ([52]) Let $P \in \mathbb{R}^{n \times n}_{\geq 0}$ be a transition matrix which is irreducible, aperiodic, and reversible. Let $\pi \in \mathbb{R}^{n}_{\geq 0}$ be the stationary distribution of $P$, and let $\lambda^* \overset{\text{def.}}{=} \max\{|\lambda_i| \mid \lambda_i \text{ is an eigenvalue of } P, |\lambda_i| < 1\}$. Let $\chi^{(T)}(T)$ denote the configuration of tokens at time $T \in \mathbb{Z}_{\geq 0}$ in a corresponding functional-router model with $M$ tokens, then

$$\left| \chi_w^{(T)} - \mu_w^{(T)} \right| \leq \sqrt{\frac{\pi(w)}{\pi_{\text{min}}} \frac{2m}{1 - \lambda^*}}$$

holds for any $w \in V$ and $T \geq 0$, where $\pi_{\text{min}} \overset{\text{def.}}{=} \min_{u \in V} \pi(u)$.

It is well known that $1/(1 - \lambda^*)$ provides (upper and lower bounds of) the mixing time of Markov chains (cf. [51]), and has been well investigated for several Markov chains, which are often reversible. Here we omit the detail of the functional router model (see [52]).

5.4. Related works on the rotor-router model

The Propp machine has been investigated intensively in recent years. Cooper and Spencer [8] dealt with a parallel-walk version of the Propp machine, and studied the discrepancy on a single vertex, meaning the difference between the number of tokens in the Propp machine and the expected number of tokens in a random walk. They showed that in the $d$-dimensional integer lattice $\mathbb{Z}^d$, the discrepancy on any node at any time is bounded by a constant $c_d$, which depends only on the dimension $d$ but is independent of the number of tokens, for any initial configurations of tokens on ‘even’ vertices and for any configurations of rotor-routers. It was later shown that $c_1$ (for the infinite path) is approximately 2.29 due to Cooper, Doerr, Spencer, and Tardos [6] (cf [12]), and $c_2$ is approximately 7.83, if every rotor-router is in clockwise (or counterclockwise) order, and 7.29 otherwise due to Doerr and Friedrich [12]. For the infinite $k$-regular tree, Cooper, Doerr, Friedrich, and Spencer [5] showed that there exist initial configurations of tokens, which are not in the same parity, and rotor-routers such that the discrepancy on a node at time $T$ is $\Omega(\sqrt{kT})$.

In contrast to those results dealing with parallel-walks on infinite graphs, Holroyd and Propp [27] analyzed a single-walk version of the rotor-router machine on finite graphs. In [27], they analyzed the hitting time of the rotor-router machine, and gave a bound that $|\nu(t)(v)/t - \pi(v)| = O(mn/t)$ on any node $v$ at any time $t$, where $\nu(t)(v)$ denotes the frequency of visits of the token on node $v$ at time $t$ in the machine, $\pi$ denotes the stationary distribution of corresponding random walk, and $n$ and $m$ denote the number of vertices and edges of the graph, respectively. Recently, Friedrich and Sauerwald [23] studied the cover time of the rotor-router machine for several basic finite graphs such as tree, star, torus, hypercube and complete graph.

As another topic on the Propp machine, the aggregation model has been investigated [40, 37, 41, 42]. For a random walk, tokens in the Internal Diffusion-Limited Aggregation (IDLA) model on $\mathbb{Z}^d$ asymptotically converge to the Euclidean ball [39], and the difference between the largest distance of a token arriving from the origin and the smallest distance of a point where no token arrived is bounded by $O(n^{1/3}\text{poly}(\log n))$ after $\Theta(n^d)$ steps [38]. For the Propp machine, Levine and Peres [40, 41, 42] showed that tokens in the rotor-router aggregation model also form the Euclidean ball, and showed several bounds for the difference between the largest distance of a token arriving from the origin and the smallest distance of a point where no token arrived. Kleber [37] gave some computational results.

Doerr, Friedrich, and Sauerwald [14] showed that information spreading by the rotor-router machine is faster than the one by a random walk on some specific graphs, namely trees with the common depth and the common degree of inner nodes, and random graphs with restricted connectivity. Doerr, Friedrich, Künnefmann, and Sauerwald [13] gave some computational results.
for this phenomena. There is much other research on information spreading by the rotor-router machine on some graphs [2, 11, 15, 16, 17, 28].

6. Concluding remarks
This article has exhibited some topics concerning randomization and derandomization of computations. Unfortunately, the author has not come to the answer for the question “What is the property that a randomized algorithm really requires for randomness?” yet. There exist several fascinating open questions, including \( \text{P} \neq \text{ZPP} \), related to the subject.

Acknowledgment
This work was/is supported in part by KAKENHI No. 23650007, 24106005 and 25700002.

References
[1] Aldous D 1983 Lect. Notes Math. 986 243–297
[2] Angelopoulos S, Doerr B, Huber A and Panagiotou K 2009 Electron. J. Comb. 16 #R102
[3] Boyer R and Moore J 1981 A fast majority vote algorithm Tech. Rep. ICSCA-CMP-32
[4] Boyer R and Moore J 1991 MJRTY—a fast majority vote algorithm Automated Reasoning: Essays in Honor of Woody Bledsoe 105–117
[5] Cooper J, Doerr B, Friedrich T and Spencer J 2010 Random Struct. Algor. 37 353–366
[6] Cooper J, Doerr B, Spencer J and Tardos G 2007 Eur. J. of Combin. 28 2072–2090
[7] Cooper C, Ilcinkas D, Klasing R and Kosowski A 2011 Distrib. Comput. 24 91–99
[8] Cooper J and Spencer J 2006 Comb. Probab. and Comput. 18 123–144
[9] Cooper J, Doerr B, Kühnemann M and Sauerwald T 2009 Proc. of ISAAC 812–821
[10] Dyer M 2011 J. Stat. Phys. 144 445–458
[11] Doerr P 2009 Introducing quasirandomness to computer science Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday 99–111
[12] Doerr B and Friedrich T 2009 Comb. Probab. and Comput. 18 123–144
[13] Doerr B, Friedrich T, Kühnemann M, and Sauerwald T 2009 Proc. of ALENEX 145–153
[14] Doerr B, Friedrich T, and Sauerwald T 2008 Proc. of SODA 773–781
[15] Doerr B, Friedrich T and Sauerwald T 2009 Proc. of ICALP 366–377
[16] Doerr B, Friedrich T, and Sauerwald T 2009 Electron. Notes Discrete Math. 34 243–247
[17] Doerr B, Huber A and Levavi A 2009 Proc. of ISAAC 812–821
[18] Dyer M 2003 Proc. of STOC 693–699
[19] Dyer M, Frieze A, and Kannan R 1991 J. ACM 38 1–17
[20] Fischer M and Salzburg S 1982 J. Algorithm 3 376–379
[21] Flajolet P 1985 BIT 25 113–134
[22] Friedrich T, Gaing M, and Sauerwald T 2010 Proc. of SODA 1620–1629
[23] Friedrich T and Sauerwald T 2010 Electron. J. Comb. 17 R167
[24] Gopalan P, Klivans A, and Meka R 2010 Polynomial-time approximation schemes for knapsack and related counting problems using branching programs arXiv:1008.3187v1
[25] Gopalan P, Klivans A, Meka R, Stefankovic D, Vempala S and Vigoda E 2011 Proc. of FOCS 817–826
[26] Häggström 2002 Finite Markov chains and algorithmic application (Cambridge University Press)
[27] Holroyd A and Propp J 2010 Rotor walks and Markov chains, Algorithmic Probability and Combinatorics (The American Mathematical Society) 105–126
[28] Huber H and Fountoulakis N 2009 Electron. Notes Discrete Math. 34 553–559
[29] Ikeda S, Kubo I and Yamashita M 2009 Theor. Comput. Sci. 410 94–100
[30] Jerrum M, Sinclair A and Vigoda E 2004 J. ACM 51 671–697
[31] Kajino H, Kijima S and Makino K 2012 Discrepancy analysis of deterministic random walks on finite irreducible graphs, Discussion paper
[32] Karp R, Shenker S and Papadimitriou C 2003 ACM T. Database Syst. 28 51–55
[33] Kijima S 2011 Proc. of HI 227–236
[34] Kijima S, Koga K and Makino K 2012 Proc. of ANALCO 16–25.
[35] Kijima S and Matsui T 2006 Random Struct. Algor. 29 243–256
[36] Kijima S and Matsui T 2008 SIAM J. Comput. 38 1484–1503
[37] Kleber M 2005 Math. Intell. 27 55–63
[38] Lawler G 1995 Ann. Probab. 23 71–86
[39] Lawler G, Bramson M and Griffeath D 1992 Ann. Probab. 20 2117–2140
[40] Levine L and Peres Y 2005 Math. Intell. 27 9–11
[41] Levine L and Peres Y 2008 Indiana U. Math. J. 57 431–450
[42] Levine L and Peres Y 2009 Potential Anal. 30 1–27
[43] Levin D, Peres Y and Wilmer E 2008 Markov Chains and Mixing Times (American Mathematical Society)
[44] Manku G and Motwani R 2002 Proc. of VLDB 346–357
[45] Matsumoto M and Nishimura T 1998 ACM T. Model. Comput. S. 8 3–30
[46] Mitzenmacher M and Upfal E 2005 Probability and Computing: Randomized Algorithms and Probabilistic Analysis (Cambridge University Press)
[47] Morris R 1978 Commun. ACM 21 840–842
[48] Morris B and Sinclair A 2004 SIAM J. Comput. 34 195–226
[49] Ogata M, Yamauchi Y, Kijima S and Yamashita M 2011 Lect. Notes Comput. Sc. 7074 514–523
[50] Propp J and Wilson D 1996 Random Struct. Algor. 9 223–252
[51] Sinclair A 1993 Algorithms for Random Generation & Counting, A Markov chain approach (Birkhäuser)
[52] Shiraga T, Yamauchi Y, Kijima S and Yamashita M 2012 Deterministic random walks for irrational transition probabilities, Discussion paper
[53] Stefankovic D, Vempala S and Vigoda E 2012 SIAM J. Comput. 41 356–366
[54] Toivonen H 1996 Proc. of VLDP 134–145
[55] Wilson D 2000 Random Struct. Algor. 16 85–113
[56] http://tcslab.csce.kyushu-u.ac.jp/~kijima/applets/2nTable/