Algorithms for Almost-uniform Generation with an Unbiased Binary Source

Ömer Eğecioğlu
Department of Computer Science
University of California
Santa Barbara, CA 93106 USA

Marcus Peinado
Institute for Algorithms
GMD National Research Center
53754 Sankt Augustin, Germany

Abstract

We consider the problem of uniform generation of random integers in the range $[1, n]$ given only a binary source of randomness. Standard models of randomized algorithms (e.g. probabilistic Turing machines) assume the availability of a random binary source that can generate independent random bits in unit time with uniform probability. This makes the task trivial if n is a power of 2. However, exact uniform generation algorithms with bounded run time do not exist if n is not a power of 2.

We analyze several *almost-uniform* generation algorithms and discuss the tradeoff between the distance of the generated distribution from the uniform distribution, and the number of operations required per random number generated. In particular, we present a new algorithm which is based on a circulant, symmetric, rapidly mixing Markov chain. For a given positive integer N, the algorithm produces an integer i in the range $[1, n]$ with probability $p_i = p_i(N)$ using $O(N \log n)$ bit operations such that $\mid p_i - 1/n \mid < c\,\beta^N$, for some constant c, where

$$\beta = \frac{2^{\frac{1}{4}}}{\pi}\left(\sqrt{2\sqrt{2} - \sqrt{5 - \sqrt{5}}}\right) \approx 0.4087.$$

This rate of convergence is superior to the estimates obtainable by commonly used methods of bounding the mixing rate of Markov chains such as conductance, direct canonical paths, and couplings.

Keywords. Random number generation, uniform distribution, Markov chain, rapid mixing, eigenvalue, circulant matrix.

1 Introduction

We consider the generation of almost-uniform random integers in the range $[1, n]$, taking into account the required time, space, and number of random bits. The basic assumption is that independent random bits can be generated in unit time. If n is an exact power of 2, say $n = 2^m$, then the generation of a uniformly distributed random integer in the range $[1, n]$ is easily accomplished in time $O(m) = O(\log n)$ by generating m consecutive random bits. However, if n is not a power of 2, no algorithm with bounded running time can generate numbers in $[1, n]$ from the exact uniform distribution (see below).

The task of generating uniformly distributed random elements of a set whose size is not an exact power of two arises frequently in the study of randomized algorithms and is usually treated as a primitive operation. This is in part justified by the fact that simple and efficient *almost-uniform* generation algorithms are known. However, it appears that the exact costs and trade-offs between accuracy and required resources of these algorithms have not been analyzed in detail. One of our aims is to explore which options exist and to compare their costs. We present two new algorithms – one based on a rapidly mixing Markov chain and one based on a reduction from approximate counting to almost uniform generation – and compare their resource requirements with those of the well-known modular algorithms.

Sinclair [18] considers the problem on an abstract level, and shows polynomial time equivalence between almost uniform generation on probabilistic Turing machines and on a different machine model which allows biased coin flips.

One important class of applications which requires uniform generators for sets of arbitrary size is the simulation of *heat bath* Markov chains (cf. [6] for a precise definition). In practice, the size of the sets can be extremely large [15]. Heat bath Markov chains are one of the standard tools in computational physics, and are used frequently in high-precision numerical simulations. It is easy to show that a bias in the distribution of the generator translates directly into a similar bias in the output distribution of the Markov chain.

We present a new algorithm which is based on the simulation of a rapidly-mixing circulant Markov chain. Its analysis gives a direct bound on the second-largest eigenvalue of the transition matrix of the Markov chain and is of interest in its own right. In particular, we observe that the commonly used methods of bounding the mixing rate of Markov chains (conductance [18], direct canonical paths [17], couplings [3]), yield weaker bounds than the one obtained here. Direct bounds on the second-largest eigenvalue of transition matrices have been obtained previously, mostly based on algebraic properties of the underlying domain (e.g. [4]). However, the structure of our Markov chain, as well as the technique used to bound its mixing rate seem different from previous results.

The probabilistic Turing machine (PTM) is the most commonly used machine model in the study of randomized algorithms [14, 18]. It is a standard Turing machine equipped with the ability to generate (or access) random bits in unit time. A PTM is deterministic, except for special coin-tossing states in which there are exactly two possible transitions, determined by the flip of an unbiased coin.

Proposition 1 *Given $n \in \mathbb{N}$ which is not a power of 2, let A_n be a randomized algorithm which outputs numbers in $[1,n]$ and whose running time is bounded by $t_n \in \mathbb{N}$. Let $r_n \le t_n$ be an upper bound on the number of random bits used by A_n. Let p_i be the probability that A_n outputs $i \in [1,n]$. There exists $i \in [1,n]$ such that*

$$|p_i - 1/n| \ge 2^{-(r_n+1)}.$$

We omit the proof due to space constraints. Intuitively, A_n has to place 2^{r_n} balls (elementary events) into n bins. If n is not a power of 2, some bins have to receive at least one ball more than others. The situation is slightly different for Las Vegas type algorithms whose run time is not bounded. In the simplest case, the algorithm can use r_n random bits, assign an equal number of elementary events to each number in $[1,n]$, and decide to use more random bits or, simply, not terminate with the remaining probability. We will concentrate on algorithms whose running time is bounded, and refer to Las Vegas type algorithms only where appropriate.

Since producing the exact uniform distribution on $[1,n]$ is not possible, we try to generate integers in $[1,n]$ with an *almost-uniform* distribution. We use the well-known *relative pointwise distance r.p.d.* (e.g. [18]) to measure the closeness of the output distribution and the uniform distribution: The r.p.d. between two probability distributions p, q on a finite set X ($q_i > 0$ for all $i \in X$) is defined as

$$\Delta(p,q) = \max_{i \in X} \frac{|p_i - q_i|}{q_i}$$

In the following, q will always be the uniform distribution, and we write $\Delta(p)$ instead of $\Delta(p,q)$ to denote the r.p.d. of p from the uniform distribution. Thus $\Delta(p) = \max_{i \in X} |np_i - 1|$.

The rest of this paper is organized as follows: In Section 2, we describe the Markov chain algorithm. Our main result on the bound of the mixing rate of the Markov chain is stated in this section. Section 3 analyzes the resource requirements of three alternative algorithms. An outline of the proof of our main theorem is given in Section 4. Remarks and conclusions are given in Section 5.

2 A Rapidly Mixing Circulant Markov Chain

In this section, we describe an algorithm based on the simulation of a rapidly mixing Markov chain. In $O(N \log n)$ time, this algorithm produces a random integer i in the range $[1, n]$ with distribution p such that

$$\Delta(p) \leq n\,\beta^N \ , \quad \text{where} \quad \beta = \frac{2^{\frac{1}{4}}}{\pi}\left(\sqrt{2\sqrt{2}-\sqrt{5-\sqrt{5}}}\right) \approx 0.4087 \ . \tag{1}$$

The bound $\beta \approx 0.4087$ deserves attention in two respects. Firstly, known algorithms reduce the r.p.d. only by a factor of 0.5 in each step. Similarly, standard methods for bounding the mixing rate of a Markov chain yield bounds which are worse than 0.5. These issues will be addressed below.

We define an $n \times n$ transition matrix $P = (\ p_{ij}\ )$ such that the corresponding Markov chain M on state space $\{1, 2, \ldots, n\}$ has the following properties: 1) M is ergodic with stationary distribution $\pi = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$; 2) M is rapidly mixing, i.e. the N-step transition matrix P^N converges quickly to the limiting probabilities; 3) M can be simulated efficiently. That is, the time to simulate one transition step is $O(\log n)$. The preprocessing time, and space requirements for M are also $O(\log n)$. Given such P, the algorithm (referred to as Algorithm I) simulates N steps of M. The first condition guarantees that M converges to the uniform distribution, and the second condition ensures that a small number N of simulation steps is sufficient. The third condition ensures that each simulation step can be executed efficiently.

An $n \times n$ *circulant matrix* $C = C(a_1, a_2, \ldots, a_n)$ is a matrix of the form

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \\ a_n & a_1 & \cdots & a_{n-1} \\ \vdots & & \vdots & \vdots \\ a_2 & a_3 & \cdots & a_1 \end{bmatrix}$$

where each row is a single right circular shift of the row above it [8].

Assume that n is not a power of 2, and let $m = \lfloor \log n \rfloor$. Then $\frac{n}{2} < 2^m < n$, and n can be written in the form $n = 2^m + p$ with $0 < p < 2^m$. Consider symmetric, circulant $n \times n$ 0-1 matrices $C = C(0, a_2, a_3, \ldots, a_n)$ where exactly 2^m of the entries $a_2, a_3, \ldots, a_n$ are equal to 1. Since we are forcing C to be symmetric, this imposes the condition $a_k = a_{n+2-k}$ for $k = 2, 3, \ldots, n$. For example, for $n = 7$, we have $m = 2$ and $p = 3$. In this case there are three such matrices: $C(0, 1, 1, 0, 0, 1, 1)$, $C(0, 1, 0, 1, 1, 0, 1)$, and $C(0, 0, 1, 1, 1, 1, 0)$. Each such matrix C defines an irreducible, aperiodic (i.e. ergodic) Markov chain M on n states $\{1, 2, \ldots, n\}$ whose transition matrix is $P = \frac{1}{2^m}\, C$. The symmetry of C guarantees that the stationary distribution of the corresponding Markov chain M is the uniform distribution on $\{1, 2, \ldots, n\}$. Note that the eigenvalues of P and C are related by a constant factor 2^m. Let $\bar{\lambda}_1$ denote the second largest eigenvalue of C. It is well-known that the mixing rate of M can be bounded by $\lambda_1 = 2^{-m}\bar{\lambda}_1$. The following inequality for the r.p.d. follows from [18, 7, 13]:

$$\Delta(p(N)) \leq n \lambda_1^N \ , \tag{2}$$

where $p(N)$ is the distribution on the states of M after N simulation steps. We consider the problem of picking the nonzero a_k's so that $\bar{\lambda}_1$ is minimized:

Theorem 1 *Suppose $n = 2^m + p$ with $0 < p < 2^m$. There exists a symmetric, circulant $n \times n$ 0-1 matrix $C^{\bar{s}} = C(0, a_2, a_3, \ldots, a_n)$ with 2^m nonzero entries in its first row such that*

$$2^{-m}\bar{\lambda}_1 \leq \frac{2^{\frac{1}{4}}}{\pi}\left(\sqrt{2\sqrt{2}-\sqrt{5-\sqrt{5}}}\right) \approx 0.4087 \ .$$

Furthermore, the first row of $C^{\bar{s}}$ contains at most two symmetrically placed blocks of 1's starting at column $\bar{s} = \lceil \frac{p}{10} \rceil + 1$.

An outline of the proof is given in Section 4. We take $M = M^{\bar{s}}$ to be the Markov chain on $\{1, 2, \ldots, n\}$ whose transition matrix is $P = P^{\bar{s}} = \frac{1}{2^m} C^{\bar{s}}$. The structure of $C^{\bar{s}}$ is such that $p_{1j} = \frac{1}{2^m}$ if and only if

$$j \in \{\bar{s} + k \mid k = 0, 1, \ldots, 2^{m-1} - 1\} \cup \{n + 2 - (\bar{s} + k) \mid k = 0, 1, \ldots, 2^{m-1} - 1\} \ . \tag{3}$$

Since P is circulant, $p_{ij} = \frac{1}{2^m}$ if and only if j is in a translate modulo n of the set of indices in (3). Thus to move from a state i of M to state j, we only need to generate a random binary number r in the range $[0, 2^m - 1]$. We then use the high order bit to select the translate of one of the two sets of consecutive indices in (3). After this, the new state j is simply the $(r+1)$-st smallest index in the subset chosen. More formally, we describe the steps of this algorithm as Algorithm I. Let RANDOM$[0, 2^m - 1]$ denote a procedure which returns a random integer r in the range $[0, 2^m - 1]$ or, equivalently, m consecutive random bits provided by our machine model (PTM).

Algorithm I :

```
    Input: n, N
    Output: i ∈ [1, n]
    begin
        m := ⌊log n⌋;  s̄ := ⌈(n − 2^m)/10⌉ + 1;
        cur_state := 1;
            for j := 1 to N do
            begin
            i := RANDOM[0, 2^m − 1] ;
            if  i ∈ [0, 2^(m−1)) then cur_state := 1 + [ (cur_state − 1 + s̄ + i) mod n ]
            else cur_state := 1 + [ (cur_state − 1 + (n + 2 − (s̄ + i − 2^(m−1))) mod n ] ;
            end
        i := cur_state;
        return(i);
    end
```

The number of operations required to take one step on the Markov chain M is $O(m) = O(\log n)$. Thus, the total running time of Algorithm I is $O(N \log n)$. By (2) and Theorem 1, after starting from an arbitrary initial state and simulating N steps of M, the probability of being in some particular state j does not differ from $1/n$ by more than a constant multiple (w.r.t. N) of β^N, where β is as given in (1).

2.1 Other methods of bounding the mixing rate

We note that the bound of $\lambda_1 \leq 0.4087$ is obtained by a rather detailed analysis (cf. Section 4), taking special properties of circulant matrices into account. The well-known general methods for estimating mixing rates, while being useful general purpose tools, appear to be too coarse-grained to yield a similar bound. We outline this in the following paragraphs. Details are omitted due to space restrictions.

The *conductance* Φ [18] which measures the expansion of the transition graph is often used to bound the second largest eigenvalue of the transition matrix via the inequality $\lambda_1 \leq 1 - \Phi^2/2$. Since, by definition, $\Phi \leq 1$, this method cannot yield a better bound than $0.5 > 0.4087$. A closer analysis for the particular case considered here shows that the conductance is significantly smaller than 1, and consequently the bound obtained in this fashion is actually much larger than 0.5.

The method of [17, 9] which bounds the second largest eigenvalue directly by a *direct canonical paths* argument (as opposed to going via the conductance) usually leads to tighter bounds than conductance-based methods. We can show by means of a counting argument that this approach does not yield a better bound than $\lambda_1 \approx 0.7$.

The *coupling* method tries to bound the mixing rate by a direct probabilistic argument and without bounding the eigenvalues. The use of the coupling method is based on [3], which bounds the mixing rate by $\mathrm{e}^{-1/(2\mathrm{e}T)}$, where T is called the coupling time. Basic but tedious steps show that $T > 2$, resulting in a mixing rate of at least $0.912 \gg 0.4087$.

3 Alternative Algorithms

In this section we analyze three alternate algorithms for the generation problem. Algorithm II and IV are straightforward modular algorithms. Algorithm III is a new algorithm and based on the reduction from almost-uniform generation to approximate counting [12].

Algorithm II: This algorithm is described in [10]: Generate a random sequence of $m = \lceil \log n \rceil$ bits. If the sequence is the binary representation of an integer i_1 in the range $[0, n-1]$, then return $i := i_1 + 1$. If not, generate another m-bit random number i_2 using the same process. If after N such trials, none of the integers $i_1, i_2, \ldots, i_N$ turns out to be in $[1, n]$, then return $i := i_N - 2^{m-1}$. A more formal description of this algorithm is given as Algorithm II.

Algorithm II :

Input: n, N
Output: $i \in [1, n]$
$m := \lceil \log n \rceil$;
  for $j := 1$ **to** N **do**
    $i :=$ RANDOM$[0, 2^m - 1]$ $+1$;
    if $i \in [1, n]$ **then return**(i) and **exit**;
  return($i - 2^{m-1}$) ;

Proposition 2 *Let $p_{II}(N)$ denote the output distribution of Alg. II. Then $\Delta(p_{II}(N)) \leq 2^{-N}$.*

Proof Omitted due to space limitations. □

Algorithm II can be run in Las Vegas mode by dropping the upper limit of N loop iterations. In this case, the expected running time $E(n)$ is

$$E(n) = \sum_{k=1}^{\infty} km \frac{n}{2^m} \left(\frac{r}{2^m}\right)^{k-1} \leq \frac{m}{(1 - \frac{r}{2^m})^2} \leq \frac{m}{(1 - \frac{1}{2})^2} \leq 8 \log n \ ,$$

where $r = 2^m - n$. Thus the expected running time of Algorithm II is no worse than $8 \log n$, independently of the parameter N. Using Chernoff bounds, it is easy to show that the running time is unlikely to exceed its expectation significantly.

Algorithm III: There is a close relation between almost-uniform generation problems and the corresponding approximate counting problems (computing the number of elements in the set) [12]. In our case, the solution to the counting problem is simply n, and the solutions of relevant subproblems are easily derived. This makes it possible to design a generation algorithm based on the well-known reduction from almost-uniform generation to approximate counting of [12].

Given a bitstring s, let $\mathrm{solns}(s) = |\{x \in [0, n-1] : \exists v : sv = x\}|$ be the number of elements of $[0, n-1]$ whose binary representation begins with s. These solutions of counting subproblems are easily computed. The algorithm generates a random element of $[0, n-1]$ one bit at a time, starting with the most significant bit. At the start of the k-th round, the $k-1$ most significant bits have been determined. The invariant is that the probability of producing any given prefix is proportional to the number of elements of $[0, n-1]$ whose most significant bits coincide with this prefix. It is easy to show by induction that this relation will hold, if the next bit is set to 1 with probability solns(prefix $\circ$ 1)/solns(prefix), where $\circ$ denotes concatenation. If, at any given point, the prefix is such that solns(prefix) $= 2^i$ (for some $i > 0$), the process can be stopped. Algorithm III summarizes these steps.

Algorithm III :

> **Input:** $n > 1$
> **Output:** $\in [1, n]$
> prefix := ϵ; $k := \lceil \log n \rceil$; (* bitlength of $(n-1)$ *)
> **repeat**
>   **if** solns(prefix $\circ$ 1) $= 0$ **then** prefix := prefix $\circ$ 0
>   **else**
>   **with probability** divide[solns(prefix $\circ$ 1), solns(prefix)], **set** prefix := prefix $\circ$ 1;
>   **otherwise set** prefix := prefix $\circ$ 0
> $k := k - 1$;
> **until** $k = 0$ or solns(prefix) is a power of 2;
> prefix := prefix $\circ 0^{k - \log \text{solns(prefix)}}$;
> **if** solns(prefix) > 1 **then** prefix := prefix$\circ$RANDOM[0, solns(prefix) $- 1$]
> **return**(prefix+1);

As stated, Algorithm III is an exact uniform generator. However, its running time is unbounded because the binary representation of $p1 := \frac{\text{solns(prefix}\circ 1)}{\text{solns(prefix)}}$ may not be finite. One obtains an approximate version of the algorithm by truncating this fraction to some finite number m of bits. Steps similar to those of [18] show that $\Delta(p_{III}) \leq 2^{-N}$ if $m \geq 2\lceil \log n \rceil + N$. Thus, achieving an accuracy of 2^{-N} requires a total of at most $2\lceil \log n \rceil^2 + N\lceil \log n \rceil$ random bits. Note that there are only $\log n$ relevant values of $p1$. Those values can be obtained and stored in a precomputation step. Thus, the algorithm needs $O(\log^2 n + N \log n)$ time and space in the worst case.

The algorithm can be run in Las Vegas mode if the probabilistic decision ('with probability ...') is implemented appropriately. Because of space restrictions, we defer the details of this part of the algorithm to the final version of the paper. For both standard and Las Vegas versions we have

Proposition 3 *The expected running time of Alg. III is* $O(\log n)$.

Again, one can use Chernoff bounds to show that the actual running time is concentrated around its expectation.

Algorithm IV: This algorithm appears to be widely used in practice: fix $m \gg \log n$, generate a random integer M in the range $[0, 2^m - 1]$ by generating m consecutive random bits, and output M mod n.

Proposition 4 *Let* $p_{IV}(m)$ *denote the output distribution of Alg. IV. Then* $\Delta(p_{IV}(m)) \leq n2^{-m}$.

Next we calculate the number of bit operations required by Algorithm IV. Let $b = \lfloor \log n \rfloor + 1$. Thus M and n are m-bit and b-bit long integers with $m \geq b$. We can consider algorithms of varying complexity for the calculation of the remainder $R = (M \mathit{mod}\ n)$ depending on the sizes of the numbers involved. Straightforward division of M by n followed by a multiplication and subtraction to calculate

Algorithm IV :

Input: n, m
Output: $i \in [1, n]$
begin
$M :=$ RANDOM$[0, 2^m - 1]$;
return$((M \bmod n) + 1)$;
end

R requires $O(mb)$ bit operations. Using the asymptotically faster Schönhage-Strassen algorithm for large integers, the integral quotient of a $2b$-bit number by a b-bit number, as well as the product of two b-bit numbers can be obtained in $O(b \log b \log \log b)$ bit operations [16, 1]. To use this algorithm for remainder calculation we prepend the binary representation of M with zeros if necessary and assume that $b-1$ divides m. Write

$$M = \sum_{i=0}^{m/(b-1)-1} M_i 2^{(b-1)i}$$

where each M_i is $b-1$ bits. The remainders $r_i = 2^{(b-1)i} \mathit{mod}\ n$ for $i = 1, 2, \ldots, m/(b-1) - 1$ can be computed by $m/(b-1)$ multiplications and divisions requiring $O(b \log b \log \log b)$ bit operations each, if the Schönhage-Strassen algorithm is used. After this phase, each quantity $M_i r_i \mathit{mod}\ n$ can be computed with an additional $O(b \log b \log \log b)$ bit operations. Finally the resulting $m/(b-1)$ remainders found are summed up modulo n in another $O(\frac{m}{b-1} b) = O(m)$ bit operations. The total number of operations required for the computation of R becomes $O(m \log b \log \log b) = O(m \log \log n \log \log \log n)$.

4 Proof of the main Theorem

In this section we outline the proof of Theorem 1.

Proof The eigenvalues of a circulant matrix $C = C(0, a_2, a_3, \ldots, a_n)$ are given by

$$\lambda_r = \sum_{k=2}^{n} a_k \omega^{(k-1)r} \ , \quad r = 0, 1, \ldots, n-1 \ ,$$

where ω is a primitive $n-$th root of unity (See, for example [8, 5]). Here we do not assume that the subscripts order the eigenvalues of C in decreasing/increasing absolute value. For $2 \le s \le \lceil \frac{p}{2} \rceil + 1$, consider the matrix C^s obtained by setting the block of entries $a_s, a_{s+1}, \ldots, a_{s+2^{m-1}-1}$, together with their mirror images in the first row equal to 1. Let $2^m \lambda_r^s$ denote the eigenvalues of C^s, $r = 1, 2, \ldots, n-1$. Thus λ_r^s are the eigenvalues of the corresponding Markov chain transition matrix P^s. By the symmetry of the a_k's, we have

$$2^m \lambda_r^s = \sum_{k=s}^{s+2^{m-1}-1} (\omega^{(k-1)r} + \bar{\omega}^{(k-1)r}) = 2 \sum_{k=0}^{2^{m-1}-1} \cos \frac{2\pi(s-1+k)r}{n} \quad ,$$

where $\bar{\omega}$ is the complex conjugate of ω. In particular $\lambda_{n-r}^s = \lambda_r^s$, $r = 1, 2, \ldots, \frac{n}{2}$. Using the trigonometric identity [11]

$$\sum_{k=0}^{h-1} \cos(a + kb) = \sin \frac{hb}{2} \ \mathrm{cosec} \frac{b}{2} \ \cos(a + \frac{h-1}{2} b) \ ,$$

with $a = \frac{2\pi(s-1)r}{n}$, $b = \frac{2\pi r}{n}$, and $h = 2^{m-1}$, we find that

$$2^m \lambda_r^s = 2\ \frac{\sin \frac{\pi r}{n} 2^{m-1}}{\sin \frac{\pi r}{n}}\ \cos \frac{\pi r}{n}(2^{m-1} + 2s - 3)\ , \qquad r = 1, 2, \ldots, \frac{n}{2}\ .$$

Let $x = \frac{\pi r}{n}$. By using the double angle formula $\sin 2x = 2 \sin x \cos x$ repeatedly, we obtain

$$\sin(2^{m-1}x) = 2^{m-1} \sin x \cos x \cos(2x) \cdots \cos(2^{m-2}x)\ .$$

Therefore,

$$\lambda_r^s = \cos x \cos(2x) \cdots \cos(2^{m-2}x)\ \cos{(2^{m-1} + 2s - 3)x}\ , \qquad r = 1, 2, \ldots, \frac{n}{2}\ . \tag{4}$$

We split the proof of the Theorem into two cases, depending on whether or not $r \geq 4$:

Case I $(r \geq 4)$ **:** For $k = 1, 2, \ldots, m-2$, let I_k denote the interval $[\frac{n}{2^{k+1}}, \frac{n}{2^k}]$. Since $2^m < n < 2^{m+1}$, the union of these intervals covers $[4, \frac{n}{2}]$. Therefore if $r \geq 4$, then $r \in I_k$ for some k.

We show that the factor $\cos 2^{(k-1)}x\ \cos 2^k x$ that appears in the product expression (4) for λ_r^s is small for $x \in [\frac{\pi}{2^{k+1}}, \frac{\pi}{2^k}]$. For a fixed $k \leq m-2$, let $t = 2^{(k-1)}x$, and $f(t) = \cos t\ \cos 2t$. Then the maximum value of f on $[\frac{\pi}{4}, \frac{\pi}{2}]$ is reached for $\cos t = \frac{1}{\sqrt{6}}$. Thus on $[\frac{\pi}{4}, \frac{\pi}{2}]$,

$$|f(t)| \leq |2\ (\frac{1}{\sqrt{6}})^3 - (\frac{1}{\sqrt{6}})\ | = \frac{2}{3\sqrt{6}}\ .$$

It follows that $|\lambda_r^s| \leq \frac{2}{3\sqrt{6}}$ for $r \in [4, \frac{n}{2}] \subseteq [\frac{n}{2^{m-1}}, \frac{n}{2}]$, independently of the value of s.

Case II $(r < 4)$ **:** Now we consider the cases $r = 1, 2, 3$. Let

$$P(r, m) = \prod_{k=0}^{m-2} \cos \frac{2^k r \pi}{n}\ .$$

Then by (4)

$$\lambda_r^s = P(r, m)\ \cos \frac{r\pi}{n}(2^{m-1} + 2s - 3)\ , \quad r = 1, 2, 3\ .$$

Since $2^m < n < 2^{m+1}$, we have $\frac{\pi}{8} < \frac{2^{m-2}\pi}{n} < \frac{\pi}{4}$. Now cosine is decreasing on $[0, \frac{\pi}{2}]$. Thus

$$\cos \frac{2^{m-2}\pi}{n} < \cos \frac{\pi}{8}\ , \quad \cos \frac{2^{m-3}\pi}{n} < \cos \frac{\pi}{16}\ , \cdots, \quad \cos \frac{\pi}{n} < \cos \frac{\pi}{2^{m+1}}\ .$$

Using the last $m-2$ of these inequalities, it follows that

$$P(1, m)\ <\ \cos \frac{2^{m-2}\pi}{n} \prod_{k=4}^{m+1} \cos \frac{\pi}{2^k}\ .$$

By Viète's formula for π

$$\prod_{k=2}^{\infty} \cos \frac{\pi}{2^k}\ =\ \frac{2}{\pi}\ ,$$

and therefore

$$\lim_{m\to\infty} \prod_{k=4}^{m} \cos \frac{\pi}{2^k}\ =\ \frac{2}{\pi \cos \frac{\pi}{4} \cos \frac{\pi}{8}}\ =\ \frac{4}{\pi\sqrt{1 + \frac{1}{\sqrt{2}}}}\ . \tag{5}$$

Since $m = \lfloor \log n \rfloor$, it follows that for large n,

$$P(1,m) \;\le\; \frac{4}{\pi\sqrt{1+\frac{1}{\sqrt{2}}}}\;\cos\frac{2^{m-2}\pi}{n} \quad .$$

Using Euler's generalization of Viète's formula in a similar fashion, we obtain the following inequalities for large n

$$P(2,m) \;\le\; \frac{2\sqrt{2}}{\pi}\;\cos\frac{2^{m-1}\pi}{n} \quad , \quad |P(3,m)| \;\le\; \frac{4\sqrt{2+\sqrt{2}}}{3\pi}\;|\cos\frac{3\cdot 2^{m-2}\pi}{n}| \quad .$$

Now let $\gamma = \frac{p}{2^m}$, $\delta = \frac{2s-3}{2^{m-1}}$. Then $0 < \gamma < 1$, $0 < \delta < \gamma$ and

$$\begin{aligned}
|\lambda_1^s| &\;\le\; \frac{4}{\pi\sqrt{1+\frac{1}{\sqrt{2}}}}\;|\cos\frac{\pi}{4}(\frac{1}{1+\gamma})\;\;\cos\frac{\pi}{2}(\frac{1+\delta}{1+\gamma})| \\
|\lambda_2^s| &\;\le\; \frac{2\sqrt{2}}{\pi}\;|\cos\frac{2\pi}{4}(\frac{1}{1+\gamma})\;\;\cos\frac{2\pi}{2}(\frac{1+\delta}{1+\gamma})| \qquad (6)\\
|\lambda_3^s| &\;\le\; \frac{4\sqrt{2+\sqrt{2}}}{3\pi}\;|\cos\frac{3\pi}{4}(\frac{1}{1+\gamma})\;\;\cos\frac{3\pi}{2}(\frac{1+\delta}{1+\gamma})|
\end{aligned}$$

The three coefficients in the inequalities (6) are about 0.9745, 0.9003, and 0.7842, respectively. In order to bound the cosine factors in (6), we choose $\delta = \frac{2}{5}\gamma$. This means that we must pick $\bar{s} = \lceil\frac{p}{10}\rceil + 1$. Since s is in the range $[2, \lceil\frac{p}{2}\rceil + 1]$, such a value $\bar{s}$ exists. If we make a change of variable by setting $t = \frac{1}{1+\gamma}$, we have $\frac{1}{2} < t < 1$, and (6) turns into

$$|\lambda_1^{\bar{s}}| \le 0.9745\; f_1(t)\;,\quad |\lambda_2^{\bar{s}}| \le 0.9003\; f_2(t)\;,\quad |\lambda_3^{\bar{s}}| \le 0.7842\; f_3(t)\;,$$

where

$$|f_1(t) = |\cos\frac{\pi}{4}t\cos\frac{\pi}{10}(3t+2)|,\;\; f_2(t) = |\cos\frac{\pi}{2}t\cos\frac{\pi}{5}(3t+2)|,\;\; f_3(t) = |\cos\frac{3\pi}{4}t\cos\frac{3\pi}{10}(3t+2)|\;.$$

It is not difficult to show that f_1 and f_3 both achieve their maximum value on $\frac{1}{2} \le t \le 1$ at the point $t = \frac{1}{2}$. Thus on $[\frac{1}{2}, 1]$, $f_1(t) \le \cos\frac{\pi}{8}\cos\frac{7\pi}{20} \approx 0.4194$ and $f_3(t) \le \cos\frac{3\pi}{8}\cos\frac{\pi}{20} \approx 0.3779$. The analysis of the behavior of f_2 is more complicated, but it can be shown that $f_2(t) \le 0.4400$ on $\frac{1}{2} \le t \le 1$ with $\bar{s} = \lceil\frac{p}{10}\rceil + 1$. This gives $|\lambda_1^{\bar{s}}| < 0.4087$, $|\lambda_2^{\bar{s}}| < 0.3961$, $|\lambda_3^{\bar{s}}| < 0.2963$.

We combine this with the previous case and find that for $r \in [1, \frac{n}{2}]$,

$$|\lambda_r^{\bar{s}}| \;\le\; \frac{4}{\pi\sqrt{1+\frac{1}{\sqrt{2}}}}\;\cos\frac{\pi}{8}\;\cos\frac{7\pi}{20}\;. \qquad (7)$$

Since $\cos\frac{2\pi}{5} = \frac{\sqrt{5}-1}{4}$, by repeated use of the half angle formula for cosine, we obtain

$$\cos\frac{7\pi}{20} = \frac{\sqrt{2\sqrt{2}-\sqrt{5-\sqrt{5}}}}{2^{\frac{5}{4}}}\;. \qquad (8)$$

Combining (8), (7), and (5), we have for $r > 0$,

$$|\,\lambda_r^{\bar{s}}\,| \le \frac{2}{\pi\cos\frac{\pi}{4}}\cos\frac{7\pi}{20} \;=\; \frac{2^{\frac{1}{4}}}{\pi}\left(\sqrt{2\sqrt{2}-\sqrt{5-\sqrt{5}}}\,\right)\;,$$

and the Theorem follows. □

5 Comparisons and Conclusions

For the algorithms presented here to generate random integers in the range $[1, n]$, the maximum relative error between the generated distribution and true uniform distribution goes to zero geometrically in all cases: with rate 0.5 for Algorithms II, III, IV (using standard division), and with rate approximately 0.4087 for Algorithm I.

The following table compares the algorithms from a different perspective. It lists the resources required by each to produce one random number with error bound 2^{-k} (r.p.d. from uniformity), based on the bounds derived in the previous sections.

Algorithm	worst case		average case	
	time	random bits	time	random bits
I	$O((k + \log n) \log n)$	$0.775(k + \log n) \log n$	cf. worst case	cf. worst case
II	$O(k \log n)$	$k \log n$	$O(\log n)$	$O(\log n)$
III	$O((k + \log n) \log n)$	$(k + 2\log n) \log n$	$O(\log n)$	$O(\log n)$
IV(simple)	$O((k + \log n) \log n)$	$k + \log n$	cf. worst case	cf. worst case
IV[16]	$O((k + \log n) \log \log n \log \log \log n)$	$k + \log n$	cf. worst case	cf. worst case

The algorithms have similar worst case running times, with Algorithm II being the fastest. The faster convergence rate of the Markov chain is hidden in the big-O notation. However, it reduces the random bits required by a factor of 0.775. Algorithm IV requires the smallest number of random bits and comes to within two bits of the lower bound of Prop. 1. Algorithms II and III can stop prematurely. Their average-case running times do not depend on k and are well below their worst-case times. This was reflected in a series of experiments we performed in which Algorithms II and III were significantly faster than Algorithms I and IV.

In the context of the construction of the Markov chain used in Algorithm I, one can address the problem of picking the best possible $n \times n$ 0-1 circulant matrix (in terms of the magnitude of the modulus of the second largest eigenvalue) for an arbitrary distribution of 1's in the first row. Theorem 1 only gives an upper bound for the modulus of the second largest eigenvalue and only in case the matrix is symmetric and the row sums are $2^{\lfloor \log n \rfloor}$. The advantage of the particular distribution of the 1's considered in Theorem 1 as blocks in the first row of the matrix is small storage and ease of transition selection for the associated Markov chain. Such a distribution, however, is not necessarily optimal for the solution of the more general problem in which constraints of space and constructibility are not critical issues.

References

[1] A. Aho, J. Hopcroft, and J. Ullman, *Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, 1974.

[2] W. Aiello, S. Rajagopalan, and R. Venkatesan. Design of Practical and Provably Good Random Number Generators. In *Proc. of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms* , 1995.

[3] D. Aldous. Random walks on finite groups and rapidly mixing Markov chains. In *Séminaire de Probabilités XVII*, Lecture Notes in Mathematics 986, pages 243–297. Springer-Verlag, 1982.

[4] N. Alon and Y. Roichman. Random Cayley graphs and expanders, 1996. Manuscript.

[5] N. Biggs. *Algebraic Graph Theory*, page 16. Cambridge University Press, 1974.

[6] R. Bubley, M. Dyer, and C. Greenhill. Beating the 2Δ bound for approximately counting colourings: A computer-assisted proof of rapid mixing. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, California, 1998.

[7] E. Çınlar. *Introduction to Stochastic Processes.* Prentice-Hall Inc., 1975.

[8] P.J. Davis. *Circulant matrices.* Wiley, 1979.

[9] P. Diaconis and D. Strook. Geometric bounds for eigenvalues of Markov chains. *Annals of Applied Probability*, 1:36–61, 1991.

[10] L. Goldschlager, E.W. Mayr, and J. Ullman. Theory of parallel computation. Unpublished Notes, 1989.

[11] I.S. Gradshteyn and I.M. Ryzhik. *Table of Integrals, Series, and Products*, page 30. Academic Press Inc., 1980.

[12] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[13] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Springer-Verlag, 1976.

[14] R. Motwani. Lecture notes on approximation algorithms. Technical report, Stanford University, 1994.

[15] M. Peinado Random generation of embedded graphs and an extension to Dobrushin uniqueness. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, Dallas, 1998.

[16] A. Schönhage and V. Strassen. "Schnelle Multiplikation grosser Zahlen", *Computing*, No. 7 (1971), 281–292.

[17] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combinatorics, Probability & Computing*, 1:351–370, 1992.

[18] A. Sinclair. *Algorithms For Random Generation And Counting.* Progress In Theoretical Computer Science. Birkhauser, Boston, 1993.