From biased coin to any discrete distribution

Karol Gryszka¹

Accepted: 8 May 2020 / Published online: 29 October 2020
© The Author(s) 2020

Abstract

In this note we construct an algorithm generating any discrete distribution with an arbitrary coin (and, as a result, with arbitrary initial distribution). The coin need not be fair and the target distribution can be supported on a countable set.

Keywords Discrete probability measure · discrete distribution · expected value · coin flip · biased coin · simulation of distribution

Mathematics Subject Classification Primary 65C10 · Secondary 60A10 · 60E05

1 Introduction

The problem of simulating specific discrete distributions is being studied by many authors and there are many ideas and approaches to that problem (see [1,5–7,9,10,13,14], also the dissertation [11] and the references therein).

Schwarz ([13]) presents a collection of ideas regarding simulation of distributions with both fair and biased coin. Von Neumann (see [9,10]) is usually indicated as the first to provide some background ideas behind that topic. Later development in this area also included other objects, such as loaded dice ([3,4]).

This article provides an algorithm that generates one of at most countably many outcomes from a given distribution. The item used to generate that is a biased coin with unknown bias. Similar approach, but for finitely many outcomes, is usually studied (see for instance [5,6,11,13]) not only from the point of view of describing algorithms, but also to test their efficiency, like average execution time (see for instance [1,5,14]). The motivation for such an algorithm is fairly simple – in order to create a random number generator one has to program a generator in such a way each number is selected with an appropriate probability. While such programming is easy for finite number of outcomes, it becomes dramatically more difficult for infinitely many (it is for instance forbidden to cut the tail from the distribution, even if the tail has low probability). And that becomes even more difficult provided numbers representing probabilities are not just fractions of integers (and thus are harder to code using binary system). Actually, as we state in Sect. 4, there is at least one algorithm that creates

✉ Karol Gryszka
karol.gryszka@up.krakow.pl

¹ Institute of Mathematics, Pedagogical University of Kraków, Podchorążych 2, 30-084 Kraków, Poland

such a simulation. However, the lack of actual source or the impossibility to find any reliable source of that made us to prepare our own approach to the problem.

In this article we recall some basic ideas, including von Neumann's algorithm and algorithms that simulate a biased coin with a fair one. Then we show how one can generate an outcome from arbitrary discrete distribution using these. Note that we are not interested in the limit process like there is one in moving from the binomial distribution to the Poisson distribution. Such an approach is useful in completely different scenario; here we search for a process that would give for instance a sample from Poisson distribution with exact probabilities and this has to exclude any limit process.

We start by recalling some basic definitions and notation. Let Ω be a non-empty set and Σ be a σ -algebra over Ω , μ be a probability measure on Σ .

A function $X: \Omega \rightarrow \mathbb{R}$ is called a *random variable* if for a given probability space (Ω, Σ, μ) and for any Borel set B we have $X^{-1}(B) \in \Sigma$. A function $P_X: \mathcal{B}(\mathbb{R}) \rightarrow [0, 1]$ defined as

$$P_X(B) = \mu(X^{-1}(B)) = \mu(\{\omega \in \Omega \mid X(\omega) \in B\})$$

is called the *distribution function of X* . Throughout the paper we use the classic convention and write $P(X = x)$ or $P(X \in B)$ for some $x \in \mathbb{R}$ and $B \in \mathcal{B}(\mathbb{R})$. Two random variables X_1 and X_2 over the same Ω and Σ are *independent* if $P((X_1, X_2) \in A \times B) = P(X_1 \in A) \cdot P(X_2 \in B)$ for any $A, B \in \Sigma$.

We use the following convention and notation. We write $D_n = \{1, 2, \dots, n\}$. If $\#\Omega = n < \aleph_0$, then we write $\Omega = \{\omega_1, \dots, \omega_n\}$ and consider random variables of the form

$$X: \Omega \ni \omega_i \mapsto i \in D_n.$$

If $\#\Omega = \aleph_0$, then we write $\Omega = \{\omega_i\}_{i \in \mathbb{N}}$ and consider random variables of the form

$$X: \Omega \ni \omega_i \mapsto i \in \mathbb{N}.$$

Here, we include $0 \in \mathbb{N}$. In either case we write $p_i := P(X = i)$ and assume $p_i > 0$. We also call such distributions discrete and call the elements of Ω *atoms*.

Our basic experiment in simulation is a coin flip, we therefore use the following identification for sides of the coin: H stands for heads and T stands for tails. We also assume that the coin cannot land on its edge with positive probability (which is actually possible, see [8]). Moreover, whenever we consider a sequence of events, we assume they are independent and, if the context is clear, they are equally distributed.

We would like to point out that we do not describe formally what does it mean to generate or simulate a distribution. This is to avoid some unnecessary and rather tedious formalization. Instead, we do it intuitively and more clearly – for instance to generate a uniform distribution over 5 atoms we think of the algorithm with 5 outcomes, each with the probability $1/5$. The classic example that we are about to describe is presented in the next section and addresses our approach.

2 The von Neumann algorithm

A coin can be biased and the bias may remain unknown. We recall the von Neumann solution that allows us to simulate a fair coin with a biased one.

Proposition 2.1 [von Neumann [9,10]] *Given any biased coin receiving heads with probability $p \in (0, 1)$, consider the following algorithm.*

1. Toss the coin twice. There are exactly four possibilities: (T, T) , (T, H) , (H, T) , (H, H) .
2. If the results are different, the first symbol is the result of the algorithm.
3. If the results are the same, go back to step 1.

Then that algorithm simulates an unbiased coin. The expected number of coin flips is $\frac{1}{p-p^2}$.

Note that we do not need assume that the bias is a known value. Still, we can determine H or T with equal chances.

The algorithm from Proposition 2.1 can be extended to other distributions supported on finite space, such as for instance one can use the similar idea to pick a natural number between 1 and 6 using a loaded dice (in that case we roll the dice until a permutation from S_6 is obtained). These algorithms are fairly simple but far from being optimal. The easiest way to improve them is to toss a coin (or roll a dice) until two (six) different faces show in consecutive draws (see [12] for the more efficient case of dice).

3 Simulation of biased coin

In this Section we briefly discuss the converse of the von Neumann's algorithm. We want to simulate biased coins using unbiased ones, which is equivalent to sampling a Bernoulli trial with the probability of success $p \in (0, 1)$. The most amazing fact that is that regardless of the bias, we can do that on average in 2 coin flips.

Let $p \in (0, 1)$ and let $[b_i]_{i \in \mathbb{N}^*}$ be the binary representation of the number p (here and later if $A \subset \mathbb{R}$, then $A^* := A \setminus \{0\}$), that is

$$p = \sum_{i=1}^{+\infty} \frac{b_i}{2^i} = [b_i]_{i \in \mathbb{N}^*}$$

(for the convenience, we slightly abuse the notation in the above identity). By the identity $0.(1) = 1$ we always assume that if the number has two expansions finite and infinite, we pick the finite one.

The following algorithms are based on [2]. We present them along with short proofs.

Proposition 3.1 ([2]) *Let $p = [b_i]_{i \in \mathbb{N}^*}$ be the probability of receiving heads in biased coin flip. Consider the following algorithm:*

1. Flip an unbiased coin until it records heads for the first time (the n -th flip).
2. If $b_n = 1$, then we call the final result heads.
3. If $b_n = 0$, then we call the final result tails.

Then that algorithm simulates the biased coin.

Proof Let B denote the event that the biased coin indicates heads. Let X be a random variable describing first appearance of heads in repeated unbiased coin flip. Then

$$P(B) = \sum_{i=1}^{+\infty} P(X = i) P(b_i = 1) = \sum_{i=1}^{+\infty} \frac{1}{2^i} \cdot b_i = p.$$

□

Proposition 3.2 ([2]) *Let $p = [b_i]_{i \in \mathbb{N}^*}$ be the probability of receiving heads in biased coin flip. Consider the following algorithm:*

1. For each flip of the coin, record 0 for heads and 1 for tails.
2. Flip as long as k -th flip records b_k .
3. Once the flip differs (the n -th flip), we call the final result heads if the n -th flip was tails and we call the final result tails if the n -th flip was heads.

Then that algorithm simulates the biased coin.

Proof Let B denote the event that the biased coin indicates heads. Let X be a random variable describing how many flips agreed with the binary expansion of p . Then

$$P(B) = \sum_{i=0}^{+\infty} P(X = i) P(b_{i+1} = 1) = \sum_{i=0}^{+\infty} \frac{1}{2^{i+1}} \cdot b_{i+1} = p.$$

□

Note that in each algorithm, the probability of termination is equal to $\frac{1}{2}$. Then the expected number of flips is the inverse of that number, that is 2. This number is optimal – we refer to [2,3] for more details.

The algorithm used in Proposition 3.2 has a simple intuitive justification. We follow the binary representation of the number until the wrong digit is recorded by the coin flip. In other words, each flip selects one of dyadic intervals, that is intervals of the form $[i \cdot 2^{-k}, (i+1) \cdot 2^{-k}]$ for some $k \geq 0$ and $0 \leq i < 2^k$. Then the final flip decides whether the number was greater or smaller than p by comparing if it belonged to the interval selected by the flip. In the next section we will use that idea to generate any discrete distribution.

4 Arbitrary distribution

In this Section we extend the idea of tracking binary representation of a number (cf. Proposition 3.2) to generate one of infinitely many outcomes. This leads to the algorithm that was suggested by the anonymous referee. The algorithm is claimed to be known, despite that we do not know who to give credit to. A search of literature revealed nothing directly related to this (described directly below) or to our (described in Theorem 4.2 and subsequent results) approach.

Let $X: \Omega \rightarrow \mathbb{R}$ be any random variable with $\#\Omega = \aleph_0$. Define the sequence $(r_i)_{i \in \mathbb{N}}$ by the formula

$$r_i = \begin{cases} 0, & i = 0, \\ r_{i-1} + p_{i-1}, & i > 0. \end{cases}$$

and define the intervals $I_i = [r_i, r_{i+1})$.

We flip the coin and record 0 for heads and 1 for tails. We record the number b_k for k -th flip. Once the n -th flip defines the binary number $b = 0.b_1b_2 \dots b_n$ that belongs to the unique interval I_m and n is the smallest number with that property (in other words, $[b, b+2^{-n}) \subset I_m$), we call the final result m .

Example 4.1 Take $p_i = 2^{-(i+1)}$ and the corresponding random variable X . Let S be the random variable defined according to the above algorithm. The sequence of flips TT gives $b = 0.11$ and sets the interval $[0.75, 1)$ that contains infinitely many intervals I_i . We therefore have to flip at least once more. Flipping H gives $b = 0.110$ that defines the interval $[0.75, 0.875)$, which is equal to I_2 . Therefore the outcome is 2. It is also clear that $P(S = i) = p_i$, that is the probability of the repeated coin flip ending at i agrees with the probability of that for X .

We find the above algorithm very interesting and simple. Our approach to this problem differs and seems more sophisticated at first glance, it actually is in some cases simpler from the point of view of theoretical calculation. Later we have found out that our idea actually generalizes the one presented in [13] (which is the special case of ours with just finitely many outcomes).

Theorem 4.2 *Let $X: \Omega \rightarrow \mathbb{R}$ be a random variable with $\#\Omega \leq \aleph_0$. There exists a coin-based algorithm that simulates X .*

Proof Let $\Omega = \{\omega_i\}_{i \in \mathbb{N}}$ and $P(X = i) = p_i > 0$ for $i \in \mathbb{N}$. Define two sequences $(p'_i)_{i \in \mathbb{N}}$ and $(q_i)_{i \in \mathbb{N}}$ by the formulas

$$p'_i = \frac{p_i}{\sum_{j \geq i} p_j}, \quad q_i = \frac{\sum_{j > i} p_j}{\sum_{j \geq i} p_j}.$$

In particular, for every $i \in \mathbb{N}$ we have $p'_i + q_i = 1$.

For $i \in \mathbb{N}$ put

$$\Omega_i = \Omega \setminus \bigcup_{j < i} \{\omega_j\}.$$

Define the sequence of independent random variables $(X_i: \Omega_i \rightarrow \mathbb{N})_{i \in \mathbb{N}}$ having the following distribution:

$$P(X_i = i) = p'_i, \quad P(X_i > i) = q_i.$$

We now describe the algorithm that generates a random variable $S: \Omega \rightarrow \mathbb{R}$ and then we check if that variable agrees with X . We proceed by induction. Our first step is to determine when $S = 0$. The probability of $X = 0$ is p_0 , and the complementary probability is q_0 . We use Proposition 3.2 to simulate a biased coin with the probability of heads equal to p_0 . We have two possibilities.

1. The biased coin flip shows heads. In that case we simulate the probability of $S = 0$.
2. The biased coin flip shows tails. In that case we simulate the probability of $S > 0$.

It is clear that $P(S = 0) = p_0 = P(X = 0)$. Note that in this case we also have $P(X = 0) = P(X_0 = 0)$.

The induction step assumes that we know how to simulate $S = n$ and we know that $P(S = i) = p_i$ for $0 \leq i \leq n$, and that the biased coin flip recorded tails in the n -th step, which translates to the simulation of $S > n$. We now show how to simulate the probability of $S = n + 1$. Note that by the induction we have eliminated all states from ω_0 to ω_n , that is we are in Ω_{n+1} and X_{n+1} is defined.

Consider a random variable X_{n+1} . We use Proposition 3.2 to simulate a biased coin with the probability of heads equal to $P(X_{n+1} = n + 1) = p'_{n+1}$. We mimic steps (1) and (2) from the description of the case $n = 0$ to define the probability of $S = n + 1$. That is, we have two possibilities.

1. The biased coin flip (defined by the X_{n+1}) shows heads. In that case we simulate the probability of $S = n + 1$.
2. The biased coin flip shows tails. In that case we simulate the probability of $S > n + 1$.

We now check if it agrees with the corresponding probability for X . By the construction of S and its relation to X_i 's,

$$\begin{aligned} P(S = n + 1) &= P(X_0 > 0, X_1 > 1, \dots, X_n > n, X_{n+1} = n + 1) \\ &= P(X_{n+1} = n + 1) \cdot \prod_{i=0}^n P(X_i > n) \\ &= p'_{n+1} \prod_{i=0}^n q_i = p'_{n+1} \prod_{i=0}^n \frac{\sum_{j>i} p_j}{\sum_{j\geq i} p_j} = p'_{n+1} \cdot \sum_{k\geq n+1} p_k = p_{n+1} \\ &= P(X = n + 1). \end{aligned}$$

Note that if Ω is finite, then the induction process terminates after $\#\Omega - 1$ many steps.

The induction shows that S and X have the same distribution. The proof is completed. $\square$

One can think of the above algorithm as weighted probability along "shorter and shorter" tails of the original distribution. It is clear that the algorithm can be generalized to select a finite set of atoms in one step instead of a single one. Then once a set is selected, we can apply that algorithm again to select a single atom.

Example 4.3 Take $p_i = 2 \cdot 3^{-(i+1)}$. Then $p'_0 = \frac{2}{3}$ and $q_0 = \frac{1}{3}$. We use the algorithm from Proposition 3.1 (or from Proposition 3.2) to simulate a biased coin with $p = \frac{2}{3}$. If the final result is H , the outcome is 0, otherwise we proceed and select one of the remaining outcomes. It is easy to check that $p'_i = \frac{2}{3}$ and $q_i = \frac{1}{3}$ for all i 's, so each subsequent step is being simulated by the same biased coin. The process terminates once the biased coin algorithm records H .

All the classic distributions supported on finitely many atoms can be simulated using the above algorithm. One can check that geometric, logarithmic and Poisson distributions are possible to simulate as well (and many more). The latter is due to simple convergence of $\lambda^k/k!$ which is in fact faster than geometric.

Corollary 4.4 Let X_1 and X_2 be any random variables supported on discrete probability spaces. Then there exists an algorithm that simulates the distribution of X_2 using the distribution of X_1 .

The key to this Corollary is that if we start with a discrete distribution (on finite or countable probability space), then we can always pick one outcome and call it H and arrange the remaining ones into a single outcome, namely T . This turns any discrete distribution into a biased coin flip.

While Theorem 4.2 and Corollary 4.4 provide an interesting result, there is one major concern with it: the series $\sum_{i=1}^{+\infty} p_i$ might converge to 1 too slow to ensure that the formula for the expected number of coin flips forms a converging series.

Proposition 4.5 Let $(n_i)_{i \in \mathbb{N}}$ be any fixed sequence of positive integers. Assume that the algorithm in Theorem 4.2 selects in a single execution n_i atoms. Then there exists a random variable X supported on a countable set of atoms such that the expected number of repetitions of the algorithm is infinite.

Proof Let $\{L_i\}_{i \in \mathbb{N}}$ be any partition of Ω with $\#L_i = n_i$. Define a random variable X as follows. Take any $\omega_j \in \Omega$. Then $\omega_j \in L_i$ for an unique $i \in \mathbb{N}$. Set

$$P(X(\omega_j)) = \frac{6}{\pi^2} \cdot \frac{1}{(i+1)^2} \cdot \frac{1}{n_i}.$$

The algorithm in Theorem 4.2 now selects one of sets L_i (instead of one atom) having the probability

$$\tilde{p}_i := P(X \in L_i) = \frac{6}{\pi^2} \cdot \frac{1}{(i+1)^2}.$$

We check that

$$\sum_{i=0}^{+\infty} \tilde{p}_i = \frac{6}{\pi^2} \cdot \sum_{i=1}^{+\infty} \frac{1}{i^2} = 1.$$

The number of repetitions required to determine L_i is by the construction equal to $i + 1$. Then, by the divergence of the harmonic sequence we have the following:

$$EY = \sum_{i=0}^{+\infty} (i+1) \tilde{p}_i = \frac{6}{\pi^2} \cdot \sum_{i=0}^{+\infty} \frac{1}{i+1} = +\infty,$$

where Y denotes the random variable describing the number of executions of the algorithm. $\square$

From the latter Proposition we immediately have the following observation: the algorithm is not universal in the sense if one wants to select a fixed number of atoms in each execution, then there always is a distribution that leads to infinitely many repetitions, on average, to complete. That of course translates to infinitely many coin flips – each selection of one set requires on average 2 coin flips.

In practice, the algorithm can fail. In order to ensure that for each distribution we can determine an outcome in finite coin flips one has to either match the sets L_i based on a given distribution (that is: based on a distribution we build the sets L_i in some way to ensure finite coin flips) or force other condition on the sequence of probabilities. We do not provide the solution to the former – we leave this as an open problem. However we provide a sufficient condition for the latter.

Theorem 4.6 *Let $X: \Omega \rightarrow \mathbb{R}$ be a random variable with $\#\Omega = \aleph_0$. Let $(p'_i)_{i \in \mathbb{N}}$ and $(q_i)_{i \in \mathbb{N}}$ be sequences defined in the proof of Theorem 4.2. If there exist $\varepsilon > 0$ and $N \geq 0$ such that $q_i < 1 - \varepsilon$ for each $i \geq N$, then the expected number of coin flips according to the algorithm in the proof of Theorem 4.2 is finite.*

Proof According to the proof of Theorem 4.2,

$$P(S = n) = p'_n \cdot \prod_{i=0}^{n-1} q_i.$$

Since each execution of the algorithm requires on average 2 coin flips (the expected number of flips as in Proposition 3.2), the expected number of coin flips EY can be estimated from above as follows:

$$\begin{aligned}
EY &= 2 \sum_{n=0}^{+\infty} \left((n+1) p'_n \cdot \prod_{i=0}^{n-1} q_i \right) \\
&= 2 \sum_{n=0}^{N-1} \left((n+1) p'_n \cdot \prod_{i=0}^{n-1} q_i \right) + 2 \sum_{n=N}^{+\infty} \left((n+1) p'_n \cdot \prod_{i=0}^{n-1} q_i \right) \\
&\leq C + 2 \sum_{n=N}^{+\infty} \left((n+1) \cdot \prod_{i=0}^{n-1} (1-\varepsilon) \right) \\
&\leq C + 2 \sum_{n=0}^{+\infty} (n+1)(1-\varepsilon)^n \\
&= C + \frac{2}{\varepsilon^2},
\end{aligned}$$

where C is an appropriate constant. The proof is completed. $\square$

The condition for the sequence $(q_i)_{i \in \mathbb{N}}$ in Theorem 4.6 can be understood as some sort of generalized approach to the geometric behaviour of the sequence $(p_i)_{i \in \mathbb{N}}$. In fact, all geometric sequences that are associated to the distribution of some random variable have $q_i = q < 1$ for all $i \in \mathbb{N}$. Also note that if Ω is a finite set, then any distribution satisfies that condition.

Theorem 4.6 requires close to geometric behaviour of the associated sequence $(q_i)_{i \in \mathbb{N}}$. However, there are obviously many other distributions that do not follow that rule. These are for instance the following.

Lemma 4.7 *If $\alpha > 1$ and $p_i = \frac{C}{(i+1)^\alpha}$ for $i \in \mathbb{N}$ and some appropriate constant C such that $\sum_{i=0}^{+\infty} p_i = 1$, then the associated sequence $(q_i)_{i \in \mathbb{N}}$ has the limit equal to 1.*

Proof We check that for $i > 0$ we have

$$\sum_{k=i}^{+\infty} \frac{1}{k^\alpha} \geq \int_i^{+\infty} \frac{1}{x^\alpha} dx = \frac{1}{(\alpha-1)i^{\alpha-1}}.$$

Then, as i goes to $+\infty$, we have

$$q_i = 1 - \frac{\frac{C}{(i+1)^\alpha}}{\sum_{k=i+1}^{+\infty} \frac{C}{k^\alpha}} \geq 1 - \frac{\frac{1}{(i+1)^\alpha}}{\frac{1}{(\alpha-1)(i+1)^{\alpha-1}}} \rightarrow 1,$$

$\square$

Example 4.8 Consider a random variable X for which p_i 's are as in the proof of Proposition 4.5. Then, by Lemma 4.7 we have $q_i \rightarrow 1$, therefore Theorem 4.6 cannot be applied to such a distribution (otherwise we would have two mutually contradicting statements).

Theorem 4.6 provides a sufficient condition for the convergence. It turns out it is not the necessary condition as we can see from the following example.

Example 4.9 Consider a random variable X for which p_i 's are defined as follows:

$$p_i := \frac{90}{\pi^4} \cdot \frac{1}{(i+1)^4}.$$

Then, as we calculate the expected number of repetition of the algorithm as in the proof of Proposition 4.5, we obtain

$$EY = \sum_{i=0}^{+\infty} (i+1)p_i = \frac{90}{\pi^4} \cdot A \approx 1.111,$$

where

$$A := \sum_{i=0}^{+\infty} \frac{1}{(i+1)^3}$$

is the Apéry's constant.

On the other hand, by Lemma 4.7 we have $q_i \rightarrow 1$.

Using Proposition 4.5, Lemma 4.7 and Example 4.9 we can easily conclude the following sufficient condition for a special class of distributions.

Theorem 4.10 *If $\alpha > 2$ and $p_i = \frac{C}{(i+1)^\alpha}$ for $i \in \mathbb{N}$ and some appropriate constant C such that $\sum_{i=0}^{+\infty} p_i = 1$, then the algorithm described in Theorem 4.2 terminates after finitely many repetitions on average.*

As a simple application of Theorem 4.10 consider the Yule-Simon distribution given by

$$f(k, \rho) = \rho B(k, \rho + 1),$$

where $\rho > 0$ is a real number, $k \geq 1$ is an integer and B is the beta function. This distribution can be simulated using a coin flip in finitely many repetitions on average provided $\rho > 1$. This is because for sufficiently large k we have

$$f(k, \rho) \propto \frac{1}{k^{\rho+1}}.$$

5 Conclusion

It turns out that all distributions (supported on finite or countable space of atoms) can be simulated using just an arbitrary coin flip. Then, a large family of them can be simulated in finite amount of time. This in particular allows us to program an actual algorithm in any language system to sample not only from built-in distributions, but from any. The algorithm itself is very simple and, comparing to the one using the binary expansion, does not force computation of all numbers involved in the distribution. This gives a huge advantage and provides a sufficient tool even for the most "extreme" cases, because only one number per the execution is required to be recalculated (the one where the algorithm stops). We leave the implementation of the algorithm to the Reader (hopefully with good programming skills).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. P. Elias, The efficient construction of an unbiased random sequence. *Ann. Math. Statist.* **43**(3), 865–870 (1972)
2. A. Irpan, *Simulating a biased coin with a fair one*, Sorta Insigful (Blog). Retrieved from: <https://www.alexirpan.com/2015/08/23/simulating-a-biased-coin-with-a-fair-one.html> (2015)
3. T. Itoh, Simulating fair dice with biased coins. *Inf. Comput.* **126**(1), 78–82 (1996)
4. A. Juels, M. Jakobsson, E. Shriver, How to turn loaded dice into fair coins. *IEEE Trans. Inf. Theory* **46**(3), 911–921 (2000)
5. D. E. Knuth, A. C.-C. Yao, *The Complexity of Nonuniform Random Number Generation*. In: J. F. Traub (eds), *Algorithms and Complexity: New Directions and Recent Results* Proceedings of a Symposium, New York, NY, Carnegie-Mellon University, Computer Science Department, pp. 357–428, Academic Press, Cambridge (1976)
6. M. C. Loui, S.-I. Pae, *Optimal random number generation from a biased coin*. In: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, p. 1079–1088 (2005)
7. M. Mitzenmacher, *Tossing a Biased Coin*. Retrieved from: <http://www.eecs.harvard.edu/~michaelm/coinflipext.pdf>
8. D. B. Murray, S.W. Teare, Probability of a tossed coin landing on edge. *Phys. Rev. E* **48**(4), 2547–2552 (1993)
9. J. von Neumann, Various techniques used in connection with random digits. *John von Neumann Collected Works* **5**, 768–770 (1963)
10. J. von Neumann, A.H. Taub, *Design of computers, theory of automata and numerical analysis* (Pergamon Press, Oxford, 1976)
11. S.-I. Pae, *Random Number Generation Using a Biased Source*. PhD dissertation, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA (2005)
12. C. Pit-Claudel, *Generating uniformly random data from skewed input: biased coins, loaded dice, skew correction, and the Von Neumann extractor*. Retrieved from: <http://pit-claudel.fr/clement/blog/generating-uniformly-random-data-from-skewed-input-biased-coins-loaded-dice-skew-correction-and-the-von-neumann-extractor/> (2012)
13. K. Schwarz, *Darts, Dice, and Coins: Sampling from a Discrete Distribution*. Retrieved from: <http://www.keithschwarz.com/darts-dice-coins/> (2011)
14. Q. F. Stout, B. Warren, Tree algorithms for unbiased coin tossing with a biased coin. *Ann. Probab.* **12**(1), 212–222 (1984)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.