Mathematics and Computers in Simulation 228 (2025) 147–155

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original articles

An asymptotically optimal algorithm for generating bin cardinalities

Luc Devroye [a,1], Dimitrios Los [b,2,*]

[a] School of Computer Science, McGill University, 3480 Rue University, QC H3A 2A7, Montreal, Canada
[b] Department of Computer Science and Technology, University of Cambridge, 15 JJ Thomson Avenue, CB3 0FD, Cambridge, United Kingdom

ARTICLE INFO

Keywords:
Random variate generation
Simulation
Expected time analysis
Balls-into-bins
Random allocations
Hashing

ABSTRACT

In the balls-into-bins setting, n balls are thrown uniformly at random into n bins. The naïve way to generate the final load vector takes $\Theta(n)$ time. However, it is well-known that this load vector has with high probability bin cardinalities of size $\Theta(\frac{\log n}{\log\log n})$. Here, we present an algorithm in the RAM model that generates the bin cardinalities of the final load vector in the optimal $\Theta(\frac{\log n}{\log\log n})$ time in expectation and with high probability.

Further, the algorithm that we present is still optimal for any $m \in [n, n\log n]$ balls and can also be used as a building block to efficiently simulate more involved load balancing algorithms. In particular, for the Two-Choice algorithm, which samples two bins in each step and allocates to the least-loaded of the two, we obtain roughly a quadratic speed-up over the naïve simulation.

1. Introduction

In the *balls-into-bins* setting, n balls are allocated uniformly at random (u.a.r.) into n bins. In this note, we present an algorithm that permits one to efficiently generate a vector of bin cardinalities. We denote by N_i the number of balls ending up in bin i. Then, we define the vector of *bin cardinalities* $(X_0, X_1, \dots, X_n)$, where

$$X_j = \sum_{i=1}^{n} \mathbf{1}_{\{N_i = j\}}, \quad 0 \le j \le n.$$

Let $K_n = \max\{j : 0 \le j \le n, X_j > 0\} = \max\{N_i : 1 \le i \le n\}$ be the *maximum occupancy*. We would like to generate $(X_0, \dots, X_{K_n}, K_n)$ in an efficient manner. As a by-product, this would yield a way of generating (X_{K_n}, K_n) which represents the number of bins with maximum occupancy, jointly with the maximum occupancy, K_n.

Throughout, we assume that we are in the idealized RAM model, in which real numbers can be stored and all standard operations on these numbers take constant time. This includes addition, multiplication, division, comparison, truncation, as well as exponentiation, logarithmic and trigonometric operations. We also have a generator at our disposal that produces an independent identically distributed (i.i.d.) sequence of Uniform$(0,1)$ random variables $U_1, U_2, \dots$, at a unit cost per U_i. In this virtual model, one can generate Binomial(n,p) and Poisson(λ) random variables in expected time uniformly bounded over all parameters n, p and λ. For references and algorithms, we refer to [1].

* Corresponding author.
E-mail addresses: lucdevroye@gmail.com (L. Devroye), mail@dimitrioslos.com (D. Los).

[1] Luc Devroye was supported by NSERC grant A3456.
[2] Dimitrios Los was supported by the LMS Early Career Fellowship (ref. 2024-19).

https://doi.org/10.1016/j.matcom.2024.08.034
Received 28 April 2024; Received in revised form 6 July 2024; Accepted 31 August 2024
Available online 2 September 2024

Just mimicking the binning process, yields a trivial linear time algorithm. However, it is known that

$$\frac{K_n}{\log n/\log\log n} \to 1$$

in probability as $n \to \infty$ [2] (cf. [3, Theorem 4.4] and [4]) and $\mathbf{E}\left[K_n^{\alpha}\right] \sim \left(\frac{\log n}{\log\log n}\right)^{\alpha}$ for all $\alpha \geq 1$ [3, Theorem 4.4]. Therefore, it is not unreasonable to hope for an algorithm with polylogarithmic expected time. We present an algorithm with $O(\log n/\log\log n)$ time in expectation and with high probability.[3] This is optimal as the output vector is w.h.p. of size $\Omega(\log n/\log\log n)$. We note that this algorithm also has optimal performance guarantees, in the case where we throw m balls into n bins, for any $n \leq m \leq n\log n$. Finally, we make use of this efficient algorithm to obtain an almost quadratic speedup for the generation of bin cardinalities for more involved balanced allocation processes. This includes the Two-Choice process, where for each ball, two bins are sampled uniformly at random and the ball is then allocated to the least loaded of the two.

Organization. In Section 2, we introduce the notation and briefly overview related work for the probability distributions and balanced allocation processes that we use. In Section 3, we present the main algorithm (Algorithm 1) for generating the bin cardinalities in optimal time (Theorem 1). In Section 4, we use this algorithm to speed up the simulation of other balanced allocation processes (Theorem 5). Finally, in Section 5, we conclude with a summary and open problems.

2. Preliminaries

Multinomial distribution. We say $X = (X_1, \ldots, X_k)$ is a Multinomial$(n; p_1, \ldots, p_k)$ distribution where $\sum_{j=1}^{k} p_j = 1$, when for all non-negative integers $n_1, \ldots, n_k$ with $\sum_{j=1}^{k} n_j = n$,

$$\mathbf{Pr}\left[\bigcap_{j=1}^{k}\{X_j = n_j\}\right] = \frac{n!}{\prod_{j=1}^{k} n_j!} \cdot \prod_{j=1}^{k} p_j^{n_j}.$$

The Multinomial$(m; 1/n, \ldots, 1/n)$ distribution corresponds to balls-into-bins with m balls and n bins. Random variates from this distribution can be generated sequentially starting with X_1, which is Binomial(n, p_1). Then X_2 is Binomial$(n - X_1, p_2)$, X_3 is Binomial$(n - X_1 - X_2, p_3)$, and so forth. As a binomial random variate can be obtained in $O(1)$ expected time, uniformly over all choices of parameters, the multinomial can thus be obtained in $O(k)$ expected time. When $k = \infty$, then the above procedure can be stopped after ℓ steps, where $\ell = \min\{j : 1 \leq j, X_1 + \cdots + X_j = n\}$, with the understanding that we return only $X_1, \ldots, X_\ell$. The time required for the generation is then equal to $O(\max\{j : 1 \leq j, X_j > 0\})$.

Multivariate hypergeometric distribution. Next, let $X = (X_1, X_2, \ldots, X_k)$ be a Hypergeometric$(n; m; n_1, n_2, \ldots, n_k)$ distribution where all parameters are positive integers, $\sum_{j=1}^{k} n_j = n$ and $m \leq n$. This corresponds to sampling m balls uniformly without replacement from an urn with n balls, n_j of which have the jth color. For $\ell_1 \leq n_1, \ell_2 \leq n_2, \ldots$, we have that

$$\mathbf{Pr}\left[\bigcap_{j=1}^{k}\{X_j = \ell_j\}\right] = \frac{\prod_{j=1}^{k}\binom{n_j}{\ell_j}}{\binom{n}{m}}.$$

Random variates from this distribution can also be generated sequentially, starting with X_1 which is Hypergeometric$(n; m; n_1, n - n_1)$. Then X_2 which is Hypergeometric$(n - n_1; m - X_1; n_2, n - n_1 - n_2)$, and so forth. As the univariate Hypergeometric$(n; m; b, n - b)$ distribution is log-concave, and the location of its mode is known to be either at $\lceil r - 1\rceil$ or at $\lfloor r\rfloor$, where $r = (m+1)(b+1)/(n+2)$, we can apply a uniformly fast generator developed by Devroye [5]. It only requires $O(1)$ time access to the value of the distribution at every point. This would require that we have an $O(1)$ time oracle at our disposal for the gamma function. However, as the rejection method is used for generating a random variate from a log-concave distribution, one can use a first-order set of Stirling bounds on the factorial,

$$\left(\frac{n}{e}\right)^n \cdot \sqrt{2\pi n} \cdot e^{\frac{1}{12n+1}} \leq n! \leq \left(\frac{n}{e}\right)^n \cdot \sqrt{2\pi n} \cdot e^{\frac{1}{12n}},$$

to accept or reject with probability close to one. If needed, we can carry out the full factorial multiplication for $n!$ in $O(n)$ time, but this is only required with probability $e^{-\Omega(n)}$, which makes the expected time to decide on exact acceptance or rejection in the rejection method $O(1)$. Some examples in this line of reasoning are given in Devroye [1]. The entire procedure for the multivariate hypergeometric takes expected time $O(k)$ and $O(k + \ell)$ time with probability at least $1 - e^{-\Omega(\ell)}$.[4] The above procedure for the multivariate hypergeometric can be stopped after ℓ steps, where $\ell = \min\{j : 1 \leq j, X_1 + \cdots + X_j = m\}$, with the understanding that we return only $X_1, \ldots, X_\ell$. Then, the time required for this is equal to $O(\max\{j : 1 \leq j, X_j > 0\})$.

Load balancing algorithms. Balls-into-bins can be used to model hashing with perfect hash functions, but it can also be used as a simple algorithm for load balancing. An improvement over this algorithm is the Two-Choice process, which is defined as follows:

[3] In this work, with high probability (w.h.p.) refers to probability at least $1 - o_n(1)$, where n is the number of bins.

[4] In [5], the concentration statement is not mentioned. However, as the algorithm is based on rejection sampling with constant acceptance probability, it follows trivially that its running time is dominated by a geometric random variable with a constant parameter, and so it has exponential tails.

Two-Choice Process:
Iteration: For each $t \geq 0$, sample *two* bins i_1 and i_2, independently and uniformly at random. Let $i \in \{i_1, i_2\}$ be such that $N_i^t = \min\{N_{i_1}^t, N_{i_2}^t\}$, breaking ties randomly. Then update:

$$N_i^{t+1} = N_i^t + 1.$$

Azar, Broder, Karlin & Upfal [6] (and Karp, Luby & Meyer auf der Heide [7]) showed that this process achieves w.h.p. a maximal occupancy of $\log_2 \log n + \Theta(1)$, when $m = n$. More generally, Berenbrink, Czumaj, Steger & Vöcking [8] showed that it achieves w.h.p. a maximal occupancy of $m/n + \log_2 \log n + \Theta(1)$, for any $m \geq n$.

In Section 4, we present an efficient algorithm for simulating Two-Choice. The algorithm also applies to a wider family of processes, namely to any process that samples two bins and uses a decision function Q to allocate to one of the two bins based on their loads. This should enable more thorough empirical comparisons of load balancing algorithms, which are commonplace in the theoretical load balancing literature (e.g., [6, Section 7], [9, Section 1.2], [10, Section 6], [11, Section 12]).

More formally, the family of Two-Sample processes is defined as follows:

Two-Sample(Q) Process:
Parameter: a *decision function* $Q : \mathbb{N} \times \mathbb{N} \to \{0, 1\}$
Iteration: For each $t \geq 0$, sample two bins i_1 and i_2, independently and uniformly at random. Then, update $N_i^{t+1} = N_i^t + 1$, where

$$i = \begin{cases} i_1 & \text{if } Q(N_{i_1}^t, N_{i_2}^t) = 0, \\ i_2 & \text{if } Q(N_{i_1}^t, N_{i_2}^t) = 1. \end{cases}$$

The Two-Choice process is an instance of this family with $Q(N_{i_1}^t, N_{i_2}^t) = \mathbf{1}_{\{N_{i_1}^t > N_{i_2}^t\}}$. Several other well-studied processes belong to this family, such as the Threshold processes [12–15], where the decision function is based on a fixed threshold f, i.e.,

$$Q_f(N_{i_1}^t, N_{i_2}^t) = \begin{cases} 0 & \text{if } N_{i_1}^t \leq f, \\ 1 & \text{otherwise.} \end{cases}$$

These Threshold processes belong to the family of Thinning processes [16], which have the several advantages of Two-Choice, the main one being that they relax the communication requirements between the two samples.

Related work. The main algorithm of this work belongs to the paradigm studied by Devroye [17] "that various random objects defined in terms of random processes can be generated quite efficiently without 'running' or 'simulating' the defining process". In that paper, the author gave efficient algorithms for generating the convex hull of a set of random points, the absorption time of a Markov chain, sums of random variables and random binary trees. Earlier examples of this paradigm include efficient generation of the maximum of n i.i.d. samples of a distribution, as in Devroye [18], or more generally their order statistics, as in Bentley & Saxe [19] and Hörmann & Derflinger [20]. Recently, Barak-Pelleg & Berend [21] introduced an efficient algorithm for sampling the coupon collector time, i.e., the time until every bin has at least one ball, that is $T = \inf\{t : 1 \leq t, \bigcap_{i=1}^n \{N_i^t \geq 1\}\}$.

For random variate generation in general, see Devroye [1] and Hörmann, Leydold & Derflinger [22]. For random variate generation for the Poisson distribution, see [23–28], for log-concave distributions, see [5,29–31] and for the Multinomial and Hypergeometric distributions, see [32–35].

For balanced allocation processes, see the books by Kolchin & Sevastyanov [36], Johnson and Kotz [37], Mahmoud [38] and surveys by Kotz & Balakrishnan [39], Mitzenmacher, Richa & Sitaraman [40] and Wieder [41].

3. Generating bin cardinalities for balls-into-bins

In this section, we present the main algorithm (Algorithm 1) for the efficient simulation of the balls-into-bins process.

Theorem 1. *For any $n \leq m \leq n \log n$, Algorithm 1 generates the vector of bin cardinalities for the* Multinomial$(m; 1/n, \ldots, 1/n)$ *distribution in time*

$$O\left(\frac{\log n}{\log\left(\frac{n}{4m} \log n\right)}\right),$$

in expectation and with probability at least $1 - o_n(1)$.

Remark 1. By the lower bound on the maximum occupancy (see e.g., [4] and [10, Lemma 14]), it follows that this time is asymptotically optimal.

Algorithm 1 GENERATE-BIN-CARDINALITIES(n, m)
The algorithm for generating the bin cardinalities X for m balls into n bins. The subroutine COMBINE-AND-SUM-CARDINALITIES(X, Y) is as outlined in Corollary 3. The parameter K^* trades-off running time and probability guarantee, see constraints in Theorem 4.

if $m \le K^*$ **then** ▷ Base case
  Add m balls u.a.r. to the bins to obtain X, using naïve simulation.
  return X
end if
$N \leftarrow 0$ ▷ Number of balls allocated
$k \leftarrow 0$ ▷ Current load
$B \leftarrow n$ ▷ Remaining bins
$\lambda \leftarrow m - m^{3/5}$ ▷ Rate of Poisson r.v.
$p \leftarrow e^{-\lambda}$ ▷ Current probability
$s \leftarrow 0$ ▷ Current cumulative probability
while $B > 0$ **do** ▷ Generation using Poissionization
  $X_k \leftarrow \mathsf{Binomial}(B, 1/(1-s))$
  $B \leftarrow B - X_k$
  $N \leftarrow N + X_k \cdot k$
  $s \leftarrow s + p$
  $p \leftarrow p \cdot \lambda/(k+1)$
  $k \leftarrow k + 1$
end while
if $N < m - 2m^{3/5}$ **then** ▷ **Case A:** We need to add many balls (*rare case*).
  Add $m - N$ balls u.a.r. to the bins in X to obtain X', using naïve simulation
  return X'
else if $N > m$ **then** ▷ **Case B:** We need to remove balls (*rare case*).
  Sample $N - m$ balls u.a.r. to remove from X to obtain X'.
  return X'
else ▷ **Case C:** We need to add few balls (*common case*).
  $Y =$ GENERATE-BIN-CARDINALITIES($n, m - N$) ▷ Proceed recursively
  return COMBINE-AND-SUM-CARDINALITIES(X, Y)
end if

We start by presenting an auxiliary algorithm for efficiently combining the bin cardinalities from two different simulations with m_x and m_y balls respectively, in order to get the bin cardinalities for $m_x + m_y$ balls. Below we prove a more general version that we also use when simulating more involved balanced allocation processes in the next section in Theorem 5.

Lemma 2. *Let X and Y be the vectors of bin cardinalities for a sample generated from a* $\mathsf{Multinomial}(m_x; 1/n, \dots, 1/n)$ *and a* $\mathsf{Multinomial}(m_y; 1/n, \dots, 1/n)$ *respectively, then we can generate the bin cardinalities for the joint distribution in $O(|X| \cdot |Y|)$ expected time and $O(|X| \cdot |Y| + \ell \cdot |X|)$ time with probability at least $1 - e^{-\ell}$ (for any $\ell \ge 1$).*

We call the subroutine for generating a sample from the joint distribution COMBINE-CARDINALITIES(X, Y).

Proof. We will construct the output vector Z, where $Z_{x,y}$ is the number of bins that were selected x times in X and y times in Y. The construction is incremental where in the rth step, we generate the entries $Z_{r,\cdot}$, i.e., those sampled r times in X, while keeping track of T, the bins not yet processed in Y. So, initially $T = Y$.

In the rth step, we sample without replacement X_r bins from the bins in T, which follows the multivariate $Z_{r,\cdot} \sim \mathsf{Hypergeometric}(\sum_i T_i; X_r; T_0, \dots, T_{|Y|-1})$ distribution. Then, we append these to Z and remove the respective counts from T. The time to complete each step is dominated by a geometric random variable with expectation $O(|Y|)$, and so overall this algorithm requires $O(|X| \cdot |Y|)$ expected time and has exponential tails, i.e., for any $\ell \ge 1$ it takes $O(|X| \cdot |Y| + \ell \cdot |X|)$ time with probability at least $1 - e^{-\ell}$ (for any $\ell \ge 1$). □

By aggregating the entries in the joint distribution with the same sum, we obtain the following corollary.

Corollary 3. *Let X and Y be the vectors of bin cardinalities for a sample generated from* $\mathsf{Multinomial}(m_x; 1/n, \dots, 1/n)$ *and* $\mathsf{Multinomial}(m_y; 1/n, \dots, 1/n)$ *independently, then we can combine them to generate the bin cardinalities of a sample from* $\mathsf{Multinomial}(m_x + m_y; 1/n, \dots, 1/n)$ *in $O(|X| \cdot |Y|)$ expected time and $O(|X| \cdot |Y| + \ell \cdot |X|)$ time with probability at least $1 - e^{-\ell}$ (for any $\ell \ge 1$).*

We call this algorithm COMBINE-AND-SUM-CARDINALITIES(X, Y) and we will use it in the proof of the main theorem below.

We are going to prove a slightly stronger version of Theorem 1, which allows us to obtain a tradeoff between the running time of the algorithm and its probability guarantee. This will be useful in Section 4 where we need a $\mathrm{polylog}(n)$ time algorithm with probability $1 - n^{-\Omega(1)}$.

Theorem 4. *Consider any $n \le m \le n \log n$ and any K^* such that*

$$10 \cdot \frac{\log n}{\log\left(\frac{4n}{m}\log n\right)} \le K^* \le 8 \cdot (\log n)^5.$$

Then, Algorithm 1 with parameter K^* generates the vector of bin cardinalities for the Multinomial$(m; 1/n, \dots, 1/n)$ *distribution in time $O(K^*)$ in expectation and with probability at least $1 - e^{-\frac{1}{4}(K^*)^{1/5}}$.*

The main theorem (Theorem 1) follows for $K^* = 10 \cdot \frac{\log n}{\log\left(\frac{4n}{m}\log n\right)}$.

Proof of Theorem 4. Let us throw N balls uniformly and independently into n bins, where N is Poisson(λ). Then, the bin sizes N_i are independent Poisson(λ/n) random variables (cf. [42, Chapter 5.1]). This implies that $(X_0, X_1, \dots)$ is Multinomial$(n; p_0, p_1, \dots)$, where p_i is the probability that one bin has size i, i.e.,

$$p_i = \mathbf{Pr}\left[\text{Poisson}(\lambda/n) = i\right], \quad i \ge 0.$$

The p_i's can be computed recursively and on the fly and we will never need more than $K = \max\{i : 1 \le 1, X_i > 0\}$ of them. This is distributed as the maximum of n independent Poisson(λ/n) random variables, which, for $\lambda = O(n)$, is bounded by $(1 + o(1)) \cdot \frac{\log n}{\log\log n}$. Thus, the Multinomial random vector $(X_0, X_1, \dots, X_K)$ can be generated in expected time $O(K)$, which is $O(\frac{\log n}{\log\log n})$. It is understood that $X_{K+i} = 0$ for all $i > 0$.

If we take $\lambda = m$, then the number of balls, N, is close to, but not equal to m. A small adjustment is required, which we present below. As the difference $|N - m|$ is in expectation $\Theta(\sqrt{m})$, it is too large to adjust point by point, repeatedly adding or removing one randomly picked element. Such a procedure would lead to an additional expected time complexity of the order of $\Theta(\sqrt{m})$, which is unacceptable.

From the Poissonized sample with $\lambda = m$, we obtain a random vector $(X_0, X_1, \dots, X_K)$, where K is distributed as the maximum of n independent Poisson(m/n) random variables. Since $\sum_{i=1}^{K} i X_i = N$, where N is Poisson(m), an adjustment may be required to correct the sample size when $N \ne m$. If there is a surplus, i.e., $N > m$, then we need to remove $N - m$ different randomly picked elements. This, however, is rather cumbersome, while adding elements is much more straightforward.

To fix this conundrum, we strategically pick $\lambda = m - m^{3/5}$.[5] The likelihood of a surplus is now much reduced, so that it is easy to remove the $N - m$ excess elements individually. First of all,

$$\begin{aligned}
\mathbf{E}\left[(N-m)_+\right] &= \sum_{j=m+1}^{\infty} j \cdot \frac{\lambda^j \cdot e^{-\lambda}}{j!} \\
&= \lambda \cdot \sum_{j=m}^{\infty} \frac{\lambda^j \cdot e^{-\lambda}}{j!} \\
&= \lambda \cdot \mathbf{Pr}\left[\text{Poisson}(\lambda) \ge m\right] \\
&\le \lambda \cdot e^{m-\lambda-m\log(m/\lambda)} \text{ (by Chernoff's bound)} \\
&= \lambda \cdot e^{m^{3/5}+m\log\left(1-\frac{m^{3/5}}{m}\right)} \\
&\le m \cdot e^{-m^{1/5}} \\
&< e^{-\frac{1}{2}m^{1/5}},
\end{aligned}$$

using Chernoff's bound [43] (cf. [44]), the inequality $\log(1-x) \le -x - x^2/2$ for any $x \in (0,1)$ and that $\lambda \le m$. By Markov's inequality, we also have that

$$\mathbf{Pr}\left[(N-m)_+ < 1\right] = 1 - \mathbf{Pr}\left[(N-m)_+ \ge 1\right] \ge 1 - e^{-\frac{1}{2}m^{1/5}}. \tag{1}$$

Similarly, we obtain that $(m - 2m^{3/5} - N)_+ < e^{-\frac{1}{2}m^{1/5}}$ in expectation and

$$\mathbf{Pr}\left[(m-2m^{3/5}-N)_+ < 1\right] \ge 1 - e^{-\frac{1}{2}m^{1/5}}. \tag{2}$$

Recall that

$$K^* \ge 10 \cdot \frac{\log n}{\log\left(\frac{4n}{m}\log n\right)}, \tag{3}$$

which bounds the maximum occupancy in expectation and with probability at least $1 - n^{-3}$ for any $n \le m \le n \log n$ (see e.g., [10, Lemma 14]).

[5] Actually, we could have chosen $\sqrt{3m\log m}$ instead of $m^{3/5}$, and still obtain the fast running time with probability at least $1 - o(1)$. But in Theorem 5, we will need probability at least $1 - n^{-\Omega(1)}$, which we obtain for a slightly different value of K^*.

Case A [$N < m - 2m^{3/5}$]: If the deficit $m - N$ is more than $2m^{3/5}$, then we add items one by one until the deficit is 0. Each addition is done first by picking a random i with probability X_i/n, decreasing X_i by one, and increasing X_{i+1} by one. Each such selection takes expected time not exceeding $O(K^*)$. Therefore the total expected time for reducing the deficit to 0 is bounded by

$$m^{3/5} \cdot \mathbf{E}\left[(m - 2m^{3/5} - N)_+\right] \cdot O(K^*) = m^{3/5} \cdot e^{-\frac{1}{2}m^{1/5}} \cdot O(K^*) = O(K^*).$$

Further, by (2), w.h.p. this case is not encountered.

Case B [$N > m$]: If on the other hand there is an excess, we remove an excess item by picking an integer i with probability iX_i/N, as this identifies a uniformly random item in a bin with cardinality i. This can be done in time not exceeding the number of choices, which in this case is in expected value (and w.h.p.) $O(K^*)$. Having chosen such an i, we decrease X_i (and thus N) by one. This is repeated until N reaches m. The total expected work for the removal procedure is thus bounded by

$$\mathbf{E}\left[(N - m)_+\right] \cdot O\left(K^*\right) = O\left(K^*\right).$$

Again, by (1), w.h.p. this case is not encountered.

Case C [$m - 2m^{3/5} \le N \le m$]: For this case, we proceed by recursively calling the algorithm with parameters $n' = n$ and $m' = m - N$, obtaining the vector $Y = (Y_0, \dots, Y_M)$. Note that $M \le 10$ with high probability, since

$$\mathbf{Pr}[\, M \ge 11 \,] \le \binom{m'}{11} \cdot \left(\frac{1}{n}\right)^{10} = O\left(\frac{(m')^{11}}{n^{10}}\right) = O\left(\frac{1}{n^3}\right),$$

using that $m' \le 2m^{3/5} \le 2(n \log n)^{3/5}$. Also, it holds that

$$\begin{aligned}
\mathbf{E}[\, M \,] &= \sum_{j=1}^{\infty} \mathbf{Pr}[\, M \ge j \,] \\
&\le 4 + \sum_{j=5}^{\infty} \binom{m'}{j} \cdot \left(\frac{1}{n}\right)^{j-1} \\
&= 4 + O(n) \cdot \sum_{j=5}^{\infty} \left(\frac{1}{n^3}\right)^{j} \\
&= 4 + o(1).
\end{aligned}$$

Using the Combine-And-Sum procedure in Corollary 3 we can combine X with Y in $O(|X| \cdot |Y|)$ expected time and $O(|X| \cdot |Y| + \ell \cdot |Y|)$ with probability at least $1 - e^{-\ell}$, for some $\ell \ge 1$ to be chosen below.

We stop the recursion once there are K^* balls left and then we naïvely simulate the remaining balls in $O(K^*)$ time.

Let r be the number of recursive calls and $m_1, \dots, m_r$ the number of balls allocated in each of these. Since $m_{i+1} \le 2m_i^{3/5}$, we have that $r = O(\log \log n)$. Further, let $T_1, \dots, T_r$ be the running time for each recursive call. The first call takes $\mathbf{E}\left[T_1\right] = O(K^*)$, the last call takes $T_k = O(K^*)$ and all others take $\mathbf{E}\left[T_i\right] = O(1)$ time. So, since $K^* \ge r$, overall the algorithm takes $O(K^*)$ time in expectation.

Now we turn to proving the high probability bound. Let $\mathcal{B}_i$ be the event that in the ith call either of the following *bad events* occur: (i) the maximum occupancy is $> K^*$, (ii) we encounter Case A or Case B or (iii) the set Y has $|Y| \ge 11$. We upper bound the probability of any of these events occurring as follows:

$$\begin{aligned}
\mathbf{Pr}\left[\bigcup_{i \in [r]} \mathcal{B}_i\right] &\le (n^{-3} + n^{-3} + 2 \cdot e^{-\frac{1}{2}m^{1/5}}) \cdot O(\log \log n) \\
&\le 3 \cdot e^{-\frac{1}{2}m^{1/5}} \cdot O(\log \log n).
\end{aligned}$$

By conditioning on none of the bad events occurring, the running times of the Combine-And-Sum procedure is dominated by a sum of independent geometric random variables. Hence, using a Chernoff bound for $\ell = \frac{1}{2}(K^*)^{1/5}$, their aggregate running time is $O(K^*)$ (constant factor times the expectation) with probability at least $1 - e^{-\frac{1}{2}(K^*)^{1/5}}$. By combining these two events, we get that the total running time is $O(K^*)$ with probability at least

$$\left(1 - 3 \cdot e^{-\frac{1}{2}m^{1/5}} \cdot O(\log \log n)\right) \cdot \left(1 - e^{-\frac{1}{2}(K^*)^{1/5}}\right) \ge 1 - e^{-\frac{1}{4}(K^*)^{1/5}},$$

using that $m \ge K^*$. □

4. Generating bin cardinalities for Two-Sample processes

In this section, we will use Algorithm 1 to speed up the simulation time of any Two-Sample process, defined in Section 2, from $\Theta(m)$ time down to $\tilde{O}(\sqrt{n})$ for any $n \le m \le n \log n$.

Theorem 5. *For any* Two-Sample(Q) *process where the decision function Q can be computed in $O(1)$ time, we can simulate the execution of the process for $n \le m \le n \log n$ balls in $O(\sqrt{n} \cdot (\log n)^6)$ time in expectation and w.h.p.*

On a high level, the proposed algorithm splits the allocations in blocks of $\Theta(\sqrt{n})$ consecutive allocations. In any such block, very few bins are sampled more than once and so most allocations can be simulated in batches. The key idea is to use Algorithm 1 to compute how many pairs of bins are sampled between any two load values ℓ_1 and ℓ_2, and then perform all these allocations in one go, as the allocated bin depends only on the values of ℓ_1 and ℓ_2. Finally, the allocations involving bins that were sampled more than once in a block can be naively simulated.

Proof. Note that for any Two-Sample process, the first and second samples are generated by a balls-into-bins process, so w.h.p. the maximum occupancy is $2 \cdot O(\log n)$. When sampling from this balls-into-bins process we will use Algorithm 1 with $K^* = 8 \cdot (\log n)^5$, so that by Theorem 4 the algorithm terminates in $O((\log n)^5)$ time with probability at least $1 - n^{-2}$.

We will split the allocations into blocks of $M = \frac{1}{4}\sqrt{n}$ balls each. The main idea is that within each block there will be a small number of bins that are sampled more than once, and so very few bins will be allocated more than one ball.

To verify this, let C_i be the indicator of the event that any of the two samples in the ith step is the same as another sample in the block. Then,

$$\mathbf{Pr}\left[C_i = 1\right] \le \frac{4M}{n} = \frac{1}{\sqrt{n}},$$

and so the number of collisions $C = \sum C_i$ in the block satisfies

$$\mathbf{E}[C] \le 2M \cdot \frac{1}{\sqrt{n}} = \frac{1}{2}.$$

Further, let C_i' be independent $\mathsf{Bernoulli}(1/\sqrt{n})$ random variables and $C' = \sum C_i'$. Then, C' stochastically dominates C and so using Chernoff's bound [43] (cf. [44]),

$$\mathbf{Pr}\left[C > 9\log n\right] \le \mathbf{Pr}\left[C' > 9\log n\right] \le n^{-2}.$$

Therefore, by taking the union bound we have that w.h.p. all blocks have at most $9\log n$ collisions (internally).

Now for each block $(t, t+M]$, we perform the following steps:

- (**Generate** $Z_{\ell,x,y}$) We generate the vector Z, where $Z_{\ell,x,y}$ counts the bins with load ℓ at step t, that were sampled x times as a first sample and y times as a second sample in the block.
  We use the hypergeometric distribution to first generate F_ℓ, the count of the number of bins with initial load ℓ that appeared as a first sample in the block. Then, using Algorithm 1 in $O((\log n)^5)$ time for each load value, we generate $F_{\ell,x}$ the number of bins with load ℓ that appeared x times as a first sample. Similarly, we generate $S_{\ell,y}$ the number of bins with load ℓ that appeared y times as a second sample. Then, we combine these two using the subroutine Combine-Cardinalities in Lemma 2, to get Z. Since there are $O(\log n)$ different load values and combining takes $O(|F| \cdot |S|)$ time, this step takes in total w.h.p. $O((\log n)^5)$ time.
- (**Pairing**) Next, we generate the pairs of samples for the allocations. For each of the special $O(\log n)$ bins which have been sampled more than once in the block, i.e., ℓ's in $Z_{\ell,x,y}$ with $x > 1$ or $y > 1$, we assign it a special bin id. Next, we generate its pairs by sampling the hypergeometric distribution, and then adjust the Z counts appropriately, giving an id to each of its paired bin (if it does not already have one). This takes $O((\log n)^5)$ time and generates a set of pairs P.
  Next, we pair the remaining items by sampling from a Hypergeometric distribution and again decreasing the counts of the sampled load values. This generates a vector P', where P'_{ℓ_1,ℓ_2} gives the count of the pairs where the first sample has load value ℓ_1 and the second load value ℓ_2 at step t. Again this takes $O((\log n)^5)$ time.
- (**Simulation**) Finally, we need to simulate the allocations. We simulate P and P' separately as there is no overlap between their bins:
  For P, sample a random order and naïvely simulate the allocations using the decision function Q. Since there are $O(\log n)$ entries in P, this takes $O(\log n)$ time.
  Next, for each pair of load values ℓ_1, ℓ_2 with $P'_{\ell_1,\ell_2} > 0$, we know that the decision function is going to allocate to the first bin if $Q(\ell_1, \ell_2) = 0$, and otherwise to ℓ_2. So, if $Q(\ell_1, \ell_2) = 0$, then there are going to be P'_{ℓ_1,ℓ_2} bins that change their load from ℓ_1 to $\ell_1 + 1$ and otherwise P'_{ℓ_1,ℓ_2} bins that change their load from ℓ_2 to $\ell_2 + 1$; so we adjust the counts accordingly. This takes $O((\log n)^2)$ time.
  In the end, in $O(\log n)$ time we aggregate the bins of the same resulting load values (using a hash table) and proceed to the next block.

Finally, we take the union bound over the $O(\sqrt{n} \cdot \log n)$ blocks having at most $O(\log n)$ collisions each and all executions of the modified Algorithm 1 terminating in $O((\log n)^5)$ time. So, we conclude that each block takes $O((\log n)^5)$ time to process and the total simulation takes w.h.p. $O(m/\sqrt{n} \cdot (\log n)^6) = O(\sqrt{n} \cdot (\log n)^6)$ time. □

The following speedup for a special case of Thinning, such as the processes studied in [16], was remarked by one of the referees.

Remark 2. Consider the special case of Thinning processes where thresholding is based on the number of times a particular bin was selected as the first sample. It is possible to simulate these processes using two rounds of balls-into-bins (i.e., in twice the time given by Theorem 1), where in the first round all the first samples are selected and in the second round all balls above the given threshold are re-allocated randomly.

5. Conclusions

In this note, we presented an algorithm in the RAM model for simulating the balls-into bins process in optimal $O(\log n/\log\log n)$ time in expectation and with high probability. The algorithm also applies to the setting with $m \in [n, n\log n]$ balls, giving optimal performance guarantees. Further, we used this algorithm to obtain a quadratic improvement in the time required to simulate a large family of load balancing processes, which includes the well-known Two-Choice process.

There are several questions that remain open. For instance, it would be interesting to extend the current algorithm for any $m > n\log n$ aiming for a time complexity of $O(\sqrt{\frac{m}{n}\cdot\log n})$, and investigate generalizations in models that are less powerful than the RAM model.

Further, it is natural to ask whether the Two-Choice process can be simulated in $\mathrm{polylog}(n)$ time, given that w.h.p. the different load values are exponentially fewer (for $m = n$) than the ones in the balls-into-bins process. Following Remark 2, it would be interesting to investigate the time to simulate other processes in the Thinning family.

CRediT authorship contribution statement

Luc Devroye: Writing – review & editing, Methodology, Formal analysis, Conceptualization. **Dimitrios Los:** Writing – review & editing, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank both referees for their wonderful suggestions.

References

[1] L. Devroye, Non-Uniform Random Variate Generation, Springer, 1986, http://dx.doi.org/10.1007/978-1-4613-8643-8.
[2] G.H. Gonnet, Expected length of the longest probe sequence in hash code searching, J. ACM 28 (2) (1981) 289–304, http://dx.doi.org/10.1145/322248.322254.
[3] L. Devroye, Lecture Notes on Bucket Algorithms, Springer Science + Business Media, LLC, 1986, http://dx.doi.org/10.1007/978-1-4899-3531-1.
[4] M. Raab, A. Steger, "Balls into bins"—a simple and tight analysis, in: 2nd International Workshop on Randomization and Computation (RANDOM'98), Vol. 1518, vol. 1518, Springer, Barcelona, Spain, 1998, pp. 159–170, http://dx.doi.org/10.1007/3-540-49543-6_13.
[5] L. Devroye, A simple generator for discrete log-concave distributions, Computing 39 (1) (1987) 87–91, http://dx.doi.org/10.1007/BF02307716.
[6] Y. Azar, A.Z. Broder, A.R. Karlin, E. Upfal, Balanced allocations, SIAM J. Comput. 29 (1) (1999) 180–200, http://dx.doi.org/10.1137/S0097539795288490.
[7] R.M. Karp, M. Luby, F. Meyer auf der Heide, Efficient PRAM simulation on a distributed memory machine, Algorithmica 16 (4–5) (1996) 517–542, http://dx.doi.org/10.1007/BF01940878.
[8] P. Berenbrink, A. Czumaj, A. Steger, B. Vöcking, Balanced allocations: the heavily loaded case, SIAM J. Comput. 35 (6) (2006) 1350–1385, http://dx.doi.org/10.1137/S009753970444435X.
[9] G. Park, A generalization of multiple choice balls-into-bins: Tight bounds, Algorithmica 77 (4) (2017) 1159–1193, http://dx.doi.org/10.1007/S00453-016-0141-Z.
[10] M. Adler, S. Chakrabarti, M. Mitzenmacher, L. Rasmussen, Parallel randomized load balancing, Random Structures Algorithms 13 (2) (1998) 159–188, http://dx.doi.org/10.1002/(SICI)1098-2418(199809)13:2<159::AID-RSA3>3.3.CO;2-Z.
[11] D. Los, T. Sauerwald, Balanced allocations with the choice of noise, J. ACM 70 (6) (2023) http://dx.doi.org/10.1145/3625386.
[12] M. Mitzenmacher, On the analysis of randomized load balancing schemes, Theory Comput. Syst. 32 (3) (1999) 361–386, http://dx.doi.org/10.1007/S002240000122.
[13] K. Iwama, A. Kawachi, Approximated two choices in randomized load balancing, in: Proceedings of 15th International Symposium on Algorithms and Computation, Vol. 3341, ISAAC'04, Springer-Verlag, 2004, pp. 545–557, http://dx.doi.org/10.1007/978-3-540-30551-4_48.
[14] D. Los, T. Sauerwald, Balanced Allocations with Incomplete Information: The Power of Two Queries, in: 13th Innovations in Theoretical Computer Science Conference (ITCS'22), Vol. 215, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022, pp. 103:1–103:23, http://dx.doi.org/10.4230/LIPIcs.ITCS.2022.103.
[15] D. Los, T. Sauerwald, J. Sylvester, Balanced allocations: Caching and packing, twinning and thinning, in: 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'22), SIAM, Alexandria, Virginia, 2022, pp. 1847–1874, http://dx.doi.org/10.1137/1.9781611977073.74.
[16] O.N. Feldheim, O. Gurel-Gurevich, The power of thinning in balanced allocation, Electron. Commun. Probab. 26 (2021) 8, http://dx.doi.org/10.1214/21-ecp400, Paper No. 34.
[17] L. Devroye, Generation of random objects, in: Proceedings of the 24th Winter Simulation Conference, Arlington, VA, USA, December 13-16, 1992, ACM Press, 1992, pp. 270–279, http://dx.doi.org/10.1145/167293.167349.
[18] L. Devroye, Generating the maximum of independent identically distributed random variables, Comput. Math. Appl. 6 (3) (1980) 305–315, http://dx.doi.org/10.1016/0898-1221(80)90039-5.
[19] J.L. Bentley, J.B. Saxe, Generating sorted lists of random numbers, ACM Trans. Math. Softw. 6 (3) (1980) 359–364, http://dx.doi.org/10.1145/355900.355907.
[20] W. Hörmann, G. Derflinger, Fast generation of order statistics, ACM Trans. Model. Comput. Simul. 12 (2) (2002) 83–93, http://dx.doi.org/10.1145/566392.566393.
[21] D. Barak-Pelleg, D. Berend, Simulating a coupon collector, in: S. Dolev, J. Katz, A. Meisels (Eds.), 6th International Symposium on Cyber Security, Cryptology, and Machine Learning, CSCML'22, in: Lecture Notes in Computer Science, vol. 13301, Springer, 2022, pp. 66–77, http://dx.doi.org/10.1007/978-3-031-07689-3_5.

[22] W. Hörmann, J. Leydold, G. Derflinger, Automatic Nonuniform Random Variate Generation, Springer Berlin, Heidelberg, 2004, http://dx.doi.org/10.1007/978-3-662-05946-3.
[23] U. Dieter, J.H. Ahrens, Acceptance-Rejection Techniques for Sampling from The Gamma and Beta Distributions, Technical Report, Stanford University, 1974.
[24] G.S. Fishman, Sampling from the Poisson distribution on a computer, Computing 17 (2) (1976) 147–156, http://dx.doi.org/10.1007/BF02276759.
[25] A.C. Atkinson, The computer generation of Poisson random variables, J. R. Stat. Soc. Ser. C. Appl. Stat. 28 (1) (1979) 29–35, http://dx.doi.org/10.2307/2346807.
[26] J.H. Ahrens, U. Dieter, Sampling from binomial and Poisson distributions: A method with bounded computation times, Computing 25 (3) (1980) 193–208, http://dx.doi.org/10.1007/BF02241999.
[27] L. Devroye, The computer generation of Poisson random variables, Computing 26 (3) (1981) 197–207, http://dx.doi.org/10.1007/BF02243478.
[28] B. Schmeiser, V. Kachitvichyanukul, Poisson Random Variate Generation, Technical Report, Purdue University, 1981.
[29] W. Hörmann, A universal generator for discrete log-concave distributions, Computing 52 (1) (1994) 89–96, http://dx.doi.org/10.1007/BF02243398.
[30] W. Hörmann, A rejection technique for sampling from T-concave distributions, ACM Trans. Math. Softw. 21 (2) (1995) 182–193, http://dx.doi.org/10.1145/203082.203089.
[31] L. Devroye, A simple algorithm for generating random variates with a log-concave density, Computing 33 (3–4) (1984) 247–257, http://dx.doi.org/10.1007/BF02242271.
[32] M.B. Brown, J. Bromberg, An efficient two-stage procedure for generating random variates from the multinomial distribution, Amer. Statist. 38 (3) (1984) 216–219, http://dx.doi.org/10.2307/2683660.
[33] E. Stadlober, Sampling from Poisson, binomial and hypergeometric distributions: ratio of uniforms as a simple and fast alternative (Ph.D. thesis), Technische Universität Graz, 1989.
[34] E. Stadlober, The ratio of uniforms approach for generating discrete random variates, J. Comput. Appl. Math. 31 (1) (1990) 181–189, http://dx.doi.org/10.1016/0377-0427(90)90349-5.
[35] C.S. Davis, The computer generation of multinomial random variates, Comput. Statist. Data Anal. 16 (2) (1993) 205–217, http://dx.doi.org/10.1016/0167-9473(93)90115-A.
[36] V.F. Kolchin, B.A. Sevastyanov, V.P. Chistyakov, Random allocations, in: Scripta Series in Mathematics, John Wiley & Sons, New York-London-Sydney, 1978, p. xi+262.
[37] N.L. Johnson, S. Kotz, Urn Models and Their Application: An Approach to Modern Discrete Probability Theory, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, New York-London-Sydney, 1977, p. xiii+402.
[38] H.M. Mahmoud, Pólya urn models, Texts in Statistical Science Series, CRC Press, Boca Raton, FL, 2009, p. xii+290.
[39] S. Kotz, N. Balakrishnan, Advances in urn models during the past two decades, in: Advances in Combinatorial Methods and Applications to Probability and Statistics, in: Stat. Ind. Technol., Birkhäuser Boston, Boston, MA, 1997, pp. 203–257, http://dx.doi.org/10.1007/978-1-4612-4140-9_14.
[40] M. Mitzenmacher, A.W. Richa, R. Sitaraman, The power of two random choices: a survey of techniques and results, in: Handbook of Randomized Computing, Vol. I, II, in: Comb. Optim., vol. 9, Kluwer Acad. Publ., Dordrecht, Netherlands, 2001, pp. 255–312, http://dx.doi.org/10.1007/978-1-4615-0013-1_9.
[41] U. Wieder, Hashing, load balancing and multiple choice, Found. Trends Theor. Comput. Sci. 12 (3–4) (2017) 275–379, http://dx.doi.org/10.1561/0400000070.
[42] M. Mitzenmacher, E. Upfal, Probability and Computing, second ed., Cambridge University Press, Cambridge, 2017, Randomization and probabilistic techniques in algorithms and data analysis.
[43] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, Ann. Math. Stat. 23 (4) (1952) 493–507, http://dx.doi.org/10.1214/aoms/1177729330.
[44] S. Boucheron, G. Lugosi, P. Massart, Concentration Inequalities - A Nonasymptotic Theory of Independence, Oxford University Press, 2013, http://dx.doi.org/10.1093/ACPROF:OSO/9780199535255.001.0001.