Random sampling of contingency tables via probabilistic divide-and-conquer

Stephen DeSalvo · James Zhao

Received: 27 June 2017 / Accepted: 23 May 2019 / Published online: 4 June 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

Abstract
We present a new approach for random sampling of contingency tables of any size and constraints based on a recently introduced probabilistic divide-and-conquer (PDC) technique. Our first application is a recursive PDC: it samples the least significant bit of each entry in the table, motivated by the fact that the bits of a geometric random variable are independent. The second application is via PDC deterministic second half, where one divides the sample space into two pieces, one of which is deterministic conditional on the other; this approach is highlighted via an exact sampling algorithm in the $2 \times n$ case. Finally, we also present a generalization to the sampling algorithm where each entry of the table has a specified marginal distribution.

Keywords Exact sampling · Approximate sampling · Transportation polytope · Boltzmann sampler

1 Introduction

1.1 Background

Let $r = (r_1, \ldots, r_m)$ and $c = (c_1, \ldots, c_n)$ be vectors of non-negative integers, with $r_1 + \cdots + r_m = c_1 + \cdots + c_n$. An $(r, c)$-contingency table is an $m \times n$ matrix $\xi = (\xi_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ with non-negative integer entries, whose row sums $r_i = \sum_j \xi_{ij}$ and column sums $c_j = \sum_i \xi_{ij}$ are prescribed by $r$ and $c$. In this paper, we propose a novel approach to random sampling from the set of $(r, c)$-contingency tables via probabilistic divide-and-conquer (PDC).
Contingency tables are an important data structure in statistics for representing the joint empirical distribution of multivariate data, and are useful for testing properties such as independence between the rows and columns (Good and Crook 1977) and similarity between two rows or two columns (Dyer and Greenhill 2000; Kijima and Matsui 2006).

Such statistical tests typically involve defining a test statistic and comparing its observed value to its distribution under the null hypothesis, that all \((r, c)\)-contingency tables are equally likely. The null distribution of such a statistic is often impossible to study analytically, but can be approximated by generating contingency tables uniformly at random.

The most popular approach in the literature for the random generation of contingency tables is Markov Chain Monte Carlo (MCMC), see, e.g., Chen et al. (2005), Cryan and Dyer (2003), Cryan et al. (2006), Diaconis and Sturmfels (1998), Fishman (2012), in which one starts with a contingency table and randomly changes a small number of entries in a way that does not affect the row sums and column sums, thereby obtaining a slightly different contingency table. After sufficiently many moves, the new table will be almost independent of the starting table; repeating this process yields almost uniform samples from the set of \((r, c)\)-contingency tables. The downside to this approach is that the number of steps one needs to perform can be quite large, see, for example, Bezáková et al. (2006), Sect. 3.1, and by the nature of MCMC, one must prescribe this number of steps before starting, so the runtime is determined not by the minimum number of steps required but the minimum provable number of steps required.

There is one Markov chain approach which yields exact samples in a finite time, namely, Markov chain coupling from the past (Propp and Wilson 1996). This approach has been successfully utilized for the random sampling of contingency tables, initially for \(2 \times n\) tables in Kijima and Matsui (2006) and extended recently to \(m \times n\) tables in Wicker (2010). The main difficulty in effectively utilizing this approach is fashioning an appropriate coupling over all possibilities states in the state space, and keeping track of when all states have coalesced. Typically, efficient couplings are formed using monotonicity, with the implication that all states are coupled once the extreme states coalesce.

Another approach is Sequential Importance Sampling (SIS) (Blitzstein and Diaconis 2011; Chen et al. 2005, 2006; Yoshida et al. 2011), where one samples from a distribution with computable deviation from uniformity, and weights samples by the inverse of their probability of occurring, to obtain unbiased estimates of any test statistic. Such techniques have proven to be quite fast, but the non-uniformity can present a problem (Bezáková et al. 2006): depending on the parameters \((r, c)\), the sample can be exponentially far from uniform, and thus the simulated distribution of the test statistic can be very different from the actual distribution despite being unbiased.

A non-Markovian approach, successfully utilized for the random sampling of contingency tables with real-valued entries in Dyer et al. (1991), associates to each state of the state space a certain parallelepiped with respect to a basis, and constructs a convex set which contains all of the parallelepipeds. The algorithm then samples uniformly from this convex set, rejecting any samples which do not lie inside one of the parallelepipeds. In Morris (2002), it is shown that this method is efficient whenever the row
Random sampling of contingency tables via probabilistic… 839

sums are each $\Omega(n^{3/2}m \log m)$ and the column sums are each $\Omega(m^{3/2}n \log n)$, where $m$ and $n$ are the number of rows and columns, respectively.

There are also extensive results pertaining to counting the number of $(r, c)$-contingency tables, see for example Baldoni-Silva et al. (2004), Barvinok (2007), Barvinok (2008), Barvinok (2009a), Barvinok (2010), Barvinok and Hartigan (2012), Barvinok et al. (2010), Bender (1974), De Loera et al. (2004), Diaconis and Gangolli (1995), Greenhill and McKay (2008), Soules (2003).

The special case of $2 \times n$ contingency tables has received particular attention in the literature, as it is relatively simple while still being interesting—many statistical applications of contingency tables involve an axis with only two categories (male/female, test/control, etc). An asymptotically uniform MCMC algorithm is presented in Dyer and Greenhill (2000). In addition, Kijima and Matsui (2006) adapted the same chain using coupling from the past to obtain an exactly uniform sampling algorithm. We describe an exactly uniform sampling algorithm for $2 \times n$ contingency tables in Sect. 4.7 whose virtue lies in its simplicity, requiring no complicated rejection functions nor lookup tables, with an analogously simple algorithm valid for real-valued tables. In addition, when all column sums are equal and all row sums are equal, its runtime is lower than its Markov chain counterparts.

1.2 Approach

The main tools we will use in this paper are rejection sampling (Von Neumann 1951) and probabilistic divide-and-conquer (PDC) (Arratia and DeSalvo 2016).

Rejection sampling is a powerful method by which one obtains random variates of a given distribution by sampling from a related distribution, and rejecting observations with probability in proportion to their likelihood of appearing in the target distribution (Von Neumann 1951). For many cases of interest, the probability of rejection is in the set $\{0, 1\}$, in which case we sample repeatedly until we obtain a sample that lies inside the target set; this is the idea behind the exact Boltzmann sampler in Duchon (2011), Duchon et al. (2004). In other applications of rejection sampling, the probability of rejection is in the interval $[0, 1]$, and depends on the outcome observed.

PDC is an exact sampling technique which appropriately pieces together samples from conditional distributions. The setup is as follows. Consider a sample space consisting of a cartesian product $A \times B$ of two probability spaces. The goal of PDC is to provide a technique to sample from the distribution of $A \times B$ restricted to some measurable event $E$. We assume that the set of objects we wish to sample from can be expressed as the collection of pairs $L((A, B) \mid (A, B) \in E)$ for some $E \subset A \times B$, where

$$A \in A, \quad B \in B \quad \text{have given distributions,}$$

$$A, B \quad \text{are independent,}$$

and either

(1) $E$ is a measurable event of positive probability; or,
(2) (i) There is some random variable $T$ on $A \times B$ which is either discrete or absolutely continuous with bounded density such that $E = \{ T = k \}$ for some $k \in \text{range}(T)$, and

(ii) For each $a \in A$, there is some random variable $T_a$ on $B$ which is either discrete or absolutely continuous with bounded density such that $\{ b \in B : (a, b) \in E \} = \{ T_a = k_a \}$ for some $k_a \in \text{range}(T_a)$.

We then sample from $L( (A, B) | (A, B) \in E )$ in two steps.

1. Generate sample from $L(A | (A, B) \in E )$, call it $x$.
2. Generate sample from $L(B | (x, B) \in E )$, call it $y$.

The PDC lemmas (Arratia and DeSalvo 2016, Lemma 2.1) in case (1), and (DeSalvo 2018, Lemma 2.1 and Lemma 2.2) in case (2), imply that the pair $(x, y)$ is an exact sample from $L( (A, B) | (A, B) \in E )$.

We champion three distinct applications of PDC, which we now describe. The first two applications start with the observation that, given row sums $r$ and columns sums $c$, collectively referred to as the line sums, the entries of a uniformly random contingency table are equal in distribution to a certain joint distribution of independent geometric random variables conditioned on satisfying the given line sums; see Sect. 3 for further details. The final application of PDC more generally applies to tables where the marginal distribution of each entry in the table is specified instead of the uniform distribution over the entire collection.

Our first application of PDC exploits a decomposition of a geometric random variable as the independent sum of a Bernoulli random variable and geometric random variable; that is, for any $0 < q < 1$, with $q$ the failure probability of a Bernoulli trial, we have

$$\text{Geometric}(q) \overset{D}{=} \text{Bernoulli} \left( \frac{q}{1+q} \right) + 2 \text{Geometric}(q^2). \quad (3)$$

We define set $A$ as the least significant bit of an entry in the table, and set $B$ as the remaining bits of that entry along with all other entries. We sample this bit according to the conditional distribution, and then repeat for each entry in the table until the parity of all entries have been sampled. The resulting partially completed table can then be reduced by a factor of 2, and sampled recursively; this is Algorithm 1. The difficulty of this approach lies in computing the conditional probability that a given entry is even/odd. For large row sums and column sums, this probability is often very nearly indistinguishable from $\frac{1}{2}$, i.e., the number of ways to complete the table assuming the given entry is even is nearly the same as assuming the given entry is odd. However, due to the recursive nature of the approach, one must inevitably encounter line sums in which the distribution of the parity of an entry is significantly asymmetric. Our proposed approximation is to treat as independent those entries not appearing in the same row or column as the given entry. The idea is similar to the strongly and weakly correlated random variables in Barvinok and Hartigan (2012, Section 4.3), where strongly correlated random variables lie in the same row/column and weakly correlated random variables are everything else. Algorithm 3 is an explicit
and efficient approximately uniform sampling algorithm which exploits as much as possible the strongly correlated random variables.

Our second application of PDC is an example of PDC deterministic second half, in which the second step is uniquely determined given the first. For $2 \times n$ tables we exploit the fact that the joint distribution of two i.i.d. geometric random variables conditional on the sum equalling a fixed target $k$ is equal in distribution to $(V, k - V)$, where $V$ is uniformly distributed over the discrete set $\{0, \ldots, k\}$. Taking set $A$ to be the first row of the last $n - 1$ columns, which can be sampled via $n - 1$ uniform random variables (each having a range from 0 to its respective column sum), and set $B$ to be the remaining entries, the completion of the table given set $A$ is unique. What makes this application particularly elegant is that the rejection function is in the constraints; otherwise we reject with probability 0 and fill in the remaining entries according to the constraints; otherwise we reject with probability 1. This is Algorithm 2. Furthermore, the algorithm generalizes naturally to real-valued tables as well, with $V$ replaced with being uniformly distributed over the interval $[0, k]$; see Algorithm 4 and Sect. 4.7.

The aforementioned results are summarized in Sect. 2. The rest of the paper is organized as follows. Section 3 describes the probabilistic interpretation of the entries of contingency tables, along with the essential PDC theory. Section 4 contains the full treatment and derivation of the algorithms described above for the random sampling of $(r, c)$-contingency tables. Section 5 formulates similar PDC algorithms for a joint distribution with given marginal probability distributions under mild restrictions.

2 Summary of main results

2.1 Definitions

Let $r = (r_1, \ldots, r_m)$ and $c = (c_1, \ldots, c_n)$ be vectors of non-negative integers, with $r_1 + \cdots + r_m = c_1 + \cdots + c_n$. Let $N = \sum_i r_i = \sum_j c_j$ be the sum of all entries. We denote the set of all $(r, c)$-contingency tables by the set

$$E \equiv E_{r,c} = \left\{ \{\xi_{ij}\}_{1 \leq i \leq m, 1 \leq j \leq n} \in \mathbb{N}_0^{m \times n} : \begin{align*}
\sum_{\ell=1}^m \xi_{i,\ell} &= r_i \quad \forall \ 1 \leq i \leq m, \\
\sum_{\ell=1}^n \xi_{\ell,j} &= c_j \quad \forall \ 1 \leq j \leq n
\end{align*} \right\}. \quad (4)$$

For a given set of row sums $r = (r_1, \ldots, r_m)$ and column sums $c = (c_1, \ldots, c_n)$, let $\Sigma(r, c)$ denote the number of $(r, c)$-contingency tables. Let $O$ denote an $m \times n$ matrix with entries in $\{0, 1\}$. For any set of row sums and column sums $(r, c)$, let $E_{r,c}(O)$ denote the set of $(r, c)$-contingency tables with entry $(i, j)$ forced to be even if the $(i, j)$th entry of $O$ is 1, and no restriction otherwise; and let $\Sigma(r, c, O)$ denote the cardinality of set $E_{r,c}(O)$. Let $O_{i,j}$ denote the matrix which has entries with value 1 in the first $j - 1$ columns, and entries with value 1 in the first $i$ rows of column $j$, and entries with value 0 otherwise.

Define for integer-valued arguments $1 \leq i \leq m, 1 \leq j \leq n, k \in \{0, 1\}$, and nonnegative vectors $r = (r_1, \ldots, r_m) \geq 0, c = (c_1, \ldots, c_n) \geq 0$, the function
We shall also consider real-valued tables with real-valued row sums and column sums, described by

\[ G_{r,c} = \left\{ \{\xi_{ij}\}_{1 \leq i \leq m, 1 \leq j \leq n} \in \mathbb{R}_{\geq 0}^{m \times n} : \begin{array}{l}
\sum_{\ell=1}^{n} \xi_{i,\ell} = r_i \forall 1 \leq i \leq m, \\
\sum_{\ell=1}^{m} \xi_{\ell,j} = c_j \forall 1 \leq j \leq n
\end{array} \right\}. \tag{5} \]

The set \( G_{r,c} \) is known as the transportation polytope, see for example Barvinok (2009b), Loera and Onn (2006). Since this set has infinite cardinality, we instead consider the density of the set with respect to Lebesgue measure on \( \mathbb{R}^{(m-1) \times (n-1)} \).

In the algorithms that follow, \( U \) will denote a uniformly distributed random variable over the interval (0, 1), independent of all other random variables, and each occurrence of \( u \) will denote a random variate from this distribution.

### 2.2 Uniform sampling of integer-valued contingency tables

Algorithm 1 below is a recursive PDC algorithm that uniformly samples from the set of nonnegative integer-valued contingency tables with any given size and row sums and column sums. Each step of the algorithm samples the least significant bit of a single entry in the table, in proportion to its prevalence under the conditional distribution, and once the least significant bit of all entries in the table have been sampled, each of the residual row sums and column sums is reduced by a factor of two. The algorithm repeats with the next significant bit, etc., until all remaining row sums and column sums are 0.
**Algorithm 1** Generation of uniformly random \((r, c)\)-contingency table

1: \(M \leftarrow \max(\max_i r_i, \max_j c_j)\).
2: \(t \leftarrow m \times n\) table with all 0 entries.
3: \(f \leftarrow \) is given by Eq. (5).
4: for \(b = 0, 1, \ldots, \lceil \log_2(M) \rceil\) do
5:  Let \(\sigma_R\) denote any permutation such that \(\sigma_R \circ r\) is in increasing order.
6:  Let \(\sigma_C\) denote any permutation such that \(\sigma_C \circ c\) is in increasing order.
7:  \(r \leftarrow \sigma_R \circ r\).
8:  \(c \leftarrow \sigma_C \circ c\).
9:  for \(j = 1, \ldots, n - 1\) do
10:     for \(i = 1, \ldots, m - 1\) do
11:        if \(u < f(i, j, 0, r, c)\) then
12:           \(\epsilon_{i,j} \leftarrow 0\)
13:        else
14:           \(\epsilon_{i,j} \leftarrow 1\)
15:        end if
16:        \(r_i \leftarrow r_i - \epsilon_{i,j}\).
17:        \(c_j \leftarrow c_j - \epsilon_{i,j}\).
18:     end for
19:     \(\epsilon_{m,j} \leftarrow c_j \mod 2\).
20:     \(c_j \leftarrow c_j - \epsilon_{m,j}\).
21:     \(c_j \leftarrow c_j / 2\).
22:     \(r_m \leftarrow r_m - \epsilon_{m,j}\).
23:  end for
24:  for \(i = 1, \ldots, m\) do
25:     \(\epsilon_{i,n} \leftarrow r_i \mod 2\).
26:     \(r_i \leftarrow r_i - \epsilon_{i,n}\).
27:     \(r_i \leftarrow r_i / 2\).
28:     \(c_n \leftarrow c_n - \epsilon_{i,n}\).
29:  end for
30:  \(c_n \leftarrow c_n / 2\).
31:  \(\epsilon \leftarrow \sigma_R^{-1} \circ \epsilon\) (Apply permutation to rows)
32:  \(\epsilon \leftarrow \sigma_C^{-1} \circ \epsilon\) (Apply permutation to columns)
33:  \(t \leftarrow t + 2^b \epsilon\).
34: end for

**Remark 1** Note that the random sampling of the least significant bits of the table at each iteration of the outer loop is not equivalent to the random sampling of binary contingency tables. The former task is considerably less constrained than the latter task, since we do not have to obtain a fixed target at each iteration.

**Theorem 1** Algorithm 1 produces a uniformly random \((r, c)\)-contingency table. It requires an expected \(O(mn \log(M))\) random bits, where \(M\) is the largest row sum or column sum.

The proof of Theorem 1 is contained in Sect. 4.1. The proof of Theorem 2 is contained in Sect. 4.2. Note that Theorem 1 is a statement about the inherent amount of randomness in the sampling algorithm, and not a complete runtime cost of Algorithm 1, which requires the computation of the non-random rejection function \(f\).

The cost to evaluate \(f\) at each iteration, or more precisely, the cost to decide Line 11, is currently the main cost of the algorithm, which requires on average the leading two bits of the evaluation of \(f\), see Knuth and Yao (1976). Fortunately, we do not always
need to evaluate the quantities exactly, and in fact typically we just need a few of the most significant bits. Thus, we may replace exact evaluation of the function \( f \) with the ability to approximate \( f \) to arbitrary precision; using, for example, an asymptotic expansion like one given in Greenhill and McKay (2008), with quantitative bounds on the error (see for example Arratia and DeSalvo 2016, Lemma 3.7) in the context of integer partitions. This motivates the following conjecture.

**Conjecture 1** Let \( \mathcal{O} \) denote an \( m \times n \) matrix with entries in \( \{ 0, 1 \} \). For any set of row sums and column sums \((r,c)\), let \( \Sigma(r,c,\mathcal{O}) \) denote the number of \((r,c)\)-contingency tables with entry \((i,j)\) forced to be even if the \((i,j)\)th entry of \( \mathcal{O} \) is 1, and no restriction otherwise. Let \( M \) denote the largest row sum or column sum. Then for any \( \mathcal{O} \), the leading \( r \) bits of \( \Sigma(r,c,\mathcal{O}) \) can be computed in \( O(2^r m^\alpha n^\beta \log^\gamma (M)) \) time, for some absolute constants \( \alpha, \beta, \gamma \geq 0 \).

**Theorem 2** Assuming Conjecture 1, Algorithm 1 is an explicit algorithm for the uniform generation of \( m \times n \) \((r,c)\)-contingency tables which is polynomial in the dimension and logarithmic in the maximal size of the line sums.

We now describe the evidence in favor of Conjecture 1. An asymptotic formula for a large class of \((r,c)\)-contingency tables is given in Barvinok and Hartigan (2012) which can be computed in time \( O(m^2 n^2) \). The approach starts with the computation of the **maximum entropy matrix** (see Barvinok and Hartigan 2010, Section 2.3), defined in terms of a polytope \( P \subseteq \mathbb{R}^n \) described via the equation \( Ax = b, x \geq 0 \), where \( A \) is a \( d \times p \) matrix with rank \( A = d < p \). This maximum entropy matrix specifies an exponential tilting of a joint distribution of independent geometric random variables which has a probability mass function which is constant on \( P \cap \mathbb{Z}^p \). The divide-and-conquer approach given in Eq. (3) shows that one would need to adapt this approach for a joint distribution of independent random variables, some of which are a geometric random variable and others which are twice a geometric random variable. The remaining work would be to adapt the saddlepoint analysis of the error given in Barvinok and Hartigan (2012). While we are optimistic that such an adaption is feasible, it is unclear in general with saddlepoint analysis whether the big-O error bounds can be made explicit, or whether the approach can be extended to provide a convergent asymptotic series expansion. (An asymptotic expansion is given in Canfield and McKay (2005, Theorem 5) for binary contingency tables with equal row sums and equal column sums which is computable in polynomial time, although because it is based on Edgeworth expansions it is not guaranteed to converge as more and more terms are summed.)

### 2.3 Approximate uniform sampling of integer-valued contingency tables

The rest of this paper demonstrates how even in the absence of Conjecture 1, one can still exploit the probabilistic interpretation of \( f \) and the PDC approach, via a natural approximation which is explicit and efficiently computable. The full details, which are somewhat lengthy, are contained in Sect. 4.4, culminating in Algorithm 3, which is essentially the same as Algorithm 1 using an approximation for \( f \) and rejection sampling for each sampled bit.
**Theorem 3** Algorithm 3 is an explicit sampling algorithm which requires on average $O(m^2 n \log M)$ random bits, with $O(m n^2 M \log^2 M)$ arithmetic operations.

While we do not at present provide a full quantitative assessment of the bias introduced via the approximation to $f$, our intention is to demonstrate how a PDC approach to solving this problem can yield immediate approximate results in the absence of complete information, and exact sampling once the appropriate numerical quantities have been fully computed.

### 2.4 Exact sampling of $2 \times n$ contingency tables

When there are only two rows, the probabilistic interpretation of the joint distribution of entries yields the following simple algorithm, which does not require the computation of any rejection functions nor a lookup table. It is an example of PDC deterministic second half.

**Algorithm 2** Generating a uniformly random $2 \times n$ $(r, c)$-contingency table.

```plaintext
1: for $j = 2, \ldots, n$ do
2:     choose $x_{1,j}$ uniformly from $\{0, \ldots, c_j\}$
3:     let $x_{2,j} = c_j - x_{1,j}$
4: end for
5: let $x_{1,1} = r_1 - \sum_{j=2}^{n} x_{1,j}$
6: let $x_{2,1} = r_2 - \sum_{j=2}^{n} x_{2,j}$
7: if $x_{1,1} < 0$ or $x_{2,1} < 0$ then
8:     restart from Line 1
9: end if
10: return $x$
```

To summarize Algorithm 2: sample entries in the top row one at a time, except the first column, uniformly between 0 and the corresponding column sum $c_j$. The rest of the table is then determined by these entries and the prescribed sums; as long as all entries produced in this way are non-negative, we accept the result.

**Theorem 4** Let $U_2, U_3, \ldots, U_n$ denote independent uniform random variables, with $U_j$ uniformly distributed over the set of integers $\{0, 1, \ldots, c_j\}$, $j = 2, \ldots, n$, and define

$$p_{r,c} := \mathbb{P}(U_2 + \cdots + U_n \in [\max(0, r_1 - c_1), r_1]).$$

(6)

Algorithm 2 produces an exact, uniformly random $2 \times n$ $(r, c)$-contingency table, with a total runtime cost which is $O(n \log(c_1)/p_{r,c})$.

In particular, when all column sums are equal, $p_{r,c}$ is simply a probability of a sum of i.i.d. uniform random variables, whose measure is known as the Irwin–Hall distribution (Hall 1927; Irwin 1927), which by the central limit theorem gives us an explicit asymptotic expression for $1/p_{r,c}$.
Corollary 1 Assuming all column sums are equal, and all row sums are equal, the total runtime cost of Algorithm 2 is $O(n^{3/2} \log(c_1))$.

We prove these and similar results for real-valued $2 \times n$ tables in Sect. 4.7, as well as describe an additional PDC deterministic second half algorithm which utilizes the maximum entropy matrix from Barvinok and Hartigan (2012) in Algorithm 5.

3 Probabilistic treatment

3.1 Rejection sampling

Let $X = (X_i)_{1 \leq i \leq s}$ denote a collection of independent random variables, $E$ some measurable set, and define the random variable $X'$ as having distribution

$$L(X') := L((X_1, \ldots, X_s) \mid E).$$

When the set $E$ has positive measure, the simplest approach to random sampling of points from the distribution (7) is to sample from $L(X)$ repeatedly until $X \in E$; this is a special case of rejection sampling (Von Neumann 1951), see also Devroye (1986), which we refer to as hard rejection sampling. The number of times we must repeat the sampling of $L(X)$ is geometrically distributed with expected value $\mathbb{P}(X \in E)^{-1}$, which may be prohibitively large. In particular, hard rejection sampling of this form fails when $\mathbb{P}(X \in E) = 0$; in this case, one must typically exploit some special structure in $L(X)$ or $E$.

The uniform distribution over many classes of decomposable combinatorial structures can be defined this way, with $X_i$ denoting the number of components of size $i$, and $E = \{\sum_{i=1}^s i X_i = s\}$ denoting the event that the random combinatorial object has weight $s$. For example, taking $X_i$ to be Geometric($1 - x^i$) for any $0 < x < 1$, Eq. (7) specifies the uniform distribution over the set of integer partitions of size $s$; taking $X_i$ to be Poisson($x^i/i!$) for any $x > 0$, Eq. (7) specifies the block sizes¹ with respect to the uniform distribution over set partitions of size $s$. See for example Arratia and Tavaré (1994), Duchon et al. (2004) and the references therein for these and further examples.

Given a collection of row sums $r = (r_1, \ldots, r_m)$, column sums $c = (c_1, \ldots, c_n)$, and set $E_{r,c}$ given in Eq. (4), the random variable $X'$, defined as having distribution

$$L(X') := L((X_{1,1}, \ldots, X_{m,n}) \mid E_{r,c}),$$

represents some measure over the set of $(r, c)$-contingency tables, with $X_{i,j}$ representing the value of the $(i, j)$th entry. In the next section we show how to choose the random variables $X_{i,j}$ in order to obtain the uniform distribution over the set of $(r, c)$-contingency tables.

¹ One must still fill those blocks, which can be done, e.g., using a permutation of $s$.  

\[ Springer \]
3.2 Uniform sampling

We now summarize some properties that we utilize to prove our main results.

Lemma 1 Suppose $X = (X_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ are independent geometric random variables with parameters $p_{ij}$. If $p_{ij}$ has the form $p_{ij} = 1 - \alpha_i \beta_j$, for $0 < \alpha_j, \beta_j < 1$, then $X$ is uniform restricted to $(r, c)$-contingency tables.

Proof For any $(r, c)$-contingency table $\xi$,

$$
\mathbb{P}(X = \xi) = \prod_{i,j} \mathbb{P}(X_{ij} = \xi_{ij}) = \prod_{i,j} (\alpha_i \beta_j)^{\xi_{ij}} (1 - \alpha_i \beta_j)
$$

$$
= \prod_{i} \alpha_i^r \prod_{j} \beta_j^c \prod_{i,j} (1 - \alpha_i \beta_j).
$$

Since this probability does not depend on $\xi$, it follows that the restriction of $X$ to $(r, c)$-contingency tables is uniform. □

For $j = 1, \ldots, n$, let $C_j = (C_{1j}, \ldots, C_{mj})$ be independent random vectors with distribution given by $(X_{1j}, \ldots, X_{mj})$ conditional on $\sum_i X_{ij} = c_j$; that is,

$$
\mathbb{P}(C_j = (\xi_{1j}, \ldots, \xi_{mj})) = \frac{\mathbb{P}(X_{1j} = \xi_{1j}, \ldots, X_{mj} = \xi_{mj})}{\mathbb{P}(\sum_i X_{ij} = c_j)}
$$

for all non-negative integer vectors $\xi_j$ with $\sum_i \xi_{ij} = c_j$, and 0 otherwise.

Lemma 2 The conditional distribution of $C = (C_1, \ldots, C_n)$ given $\sum_j C_{ij} = r_i$ for all $i$ is that of a uniformly random $(r, c)$-contingency table.

Proof For any $(r, c)$-contingency table $\xi$, $\mathbb{P}(C = \xi)$ is a constant multiple of $\mathbb{P}(X = \xi)$. □

Our next lemma demonstrates the existence of a simple and explicit formula for the parameters $p_{ij}$ which maintains the uniform distribution while at the same time optimizing the probability of generating a point inside of $E$.

Lemma 3 Suppose $X$ is a table of independent geometric random variables, where $X_{i,j}$ has parameter $p_{ij} = m/(m + c_j)$, $1 \leq i \leq m, 1 \leq j \leq n$. Then the expected columns sums of $X$ are $c$, and the expected row sums of $X$ are $N/m$.

Proof For any $j = 1, \ldots, n$,

$$
\sum_{i=1}^{m} \mathbb{E}[x_{ij}] = \sum_{i=1}^{m} \left( \frac{m + c_j}{m} - 1 \right) = c_j.
$$
Similarly, for any \( i = 1, \ldots, m \),
\[
\sum_{j=1}^{n} \mathbb{E}[x_{ij}] = \sum_{j=1}^{n} \left( \frac{m + c_j}{m} - 1 \right) = \frac{N}{m}.
\]
\[\square\]

**Remark 2**  Note that entries in different rows (columns, resp.) are conditionally independent. This means that we can separately sample all of the rows (columns, resp.) independently until all of the row (column, resp.) conditions are satisfied. It also means that once all but one column (row, resp.) are sampled according to the appropriate conditional distribution, the remaining column (row, resp.) is uniquely determined by the constraints.

For real-valued tables, the calculations are analogous and straightforward.

**Lemma 4**  Suppose \( X = (X_{ij})_{1 \leq i \leq m, 1 \leq j \leq n} \) are independent exponential random variables with parameters \( \lambda_{i,j} := -\log(1 - p_{ij}) \). If \( p_{ij} \) has the form \( p_{ij} = 1 - \alpha_i \beta_j \), then \( X \) is uniform restricted to real-valued \((r, c)\)-contingency tables.

**Proof**  For any real-valued \((r, c)\)-contingency table \( \xi \),
\[
\mathbb{P}(X \in d\xi) = \prod_{i,j} \mathbb{P}(X_{ij} \in d\xi_{ij}) = \prod_{i,j} \lambda_{i,j} e^{-\lambda_{i,j} \xi_{i,j}}
\]
\[
= \prod_{i,j} (\lambda_{i,j}) \prod_{i,j} (\alpha_i \beta_j)^{\xi_{i,j}} = \prod_{i} \alpha_i^{r_i} \prod_{j} \beta_j^{c_j} \prod_{i,j} (\lambda_{i,j}).
\]
Since this probability does not depend on \( \xi \), it follows that the restriction of \( X \) to real-valued \((r, c)\)-contingency tables is uniform. \[\square\]

**Lemma 5**  Suppose \( X \) is a table of independent exponential random variables, where \( X_{i,j} \) has parameter \( \lambda_{i,j} = -\log(1 - p_{ij}) \), \( 1 \leq i \leq m, 1 \leq j \leq n \). Then the expected columns sums of \( X \) are \( c \), and the expected row sums of \( X \) are \( \bar{N}/m \).

**Remark 3**  Remark 2 applies when the \( X_{i,j} \)'s are independent and have any discrete distribution, not just geometric. If the \( X_{i,j} \)'s are independent continuous random variables such that each of the sums \( \sum_{i=1}^{m} X_{i,j} \) for \( j = 1, 2, \ldots, n \) and \( \sum_{j=1}^{n} X_{i,j} \) for \( i = 1, 2, \ldots, m \) has a density, then the same conditional independence of entries in different rows are conditionally independent, and we can separately sample all of the rows independently until all of the row conditions are satisfied.

**Remark 4**  An alternative choice of parameters \( p_{ij} \) is developed in Barvinok (2009b), which we now briefly describe. Let \( g(x) = (x + 1) \ln(x + 1) - x \ln x \). For any given row sums and column sums \( r, c \), consider the set \( E_{r,c} \) of all integer-valued \((r, c)\)-contingency tables. For any \( X = (x_{i,j})_{i,j} \in E_{r,c} \), the function \( g(X) = \sum_{i,j} g(x_{i,j}) \) is strictly log-concave, and hence assumes a unique maximum on \( E_{r,c} \), say \( (z_{i,j})_{i,j} \), known as the maximum entropy matrix, easily found by standard methods. Barvinok (2009b, Theorem (1.7)) demonstrates that choosing parameters \( p_{ij} = (1 + z_{i,j})^{-1} \) for all \( i, j \) results in the uniform distribution over \( E_{r,c} \).
3.3 Probabilistic divide-and-conquer

Suppose that $\mathcal{A}$ and $\mathcal{B}$ are sets, and let $E$ be a subset of $\mathcal{A} \times \mathcal{B}$. Let $A$ and $B$ be probability measures on $\mathcal{A}$ and $\mathcal{B}$, respectively.

**Theorem 5** (Probabilistic divide-and-conquer (Arratia and DeSalvo 2016)) For each $a \in \mathcal{A}$, let $\mathcal{B}_a = \{b \in \mathcal{B} : (a, b) \in E\}$, where $\mathbb{P}((A, B) \in E) > 0$. Let

$$
\mathcal{L}(A') := \mathcal{L}(A \mid (A, B) \in E), \\
\mathcal{L}(B'_a) := \mathcal{L}(B \mid \mathcal{B}_a).
$$

Then, $\mathcal{L}(A', B'_a) = \mathcal{L}((A, B) \mid E)$.

A similar theorem holds when $\mathbb{P}((A, B) \in E) = 0$, under some simple conditions.

**Theorem 6** (Probabilistic divide-and-conquer for densities (DeSalvo 2018, Lemma 3.1)) For each $a \in \mathcal{A}$, let $\mathcal{B}_a = \{b \in \mathcal{B} : (a, b) \in E\}$, where $\mathbb{P}((A, B) \in E) = 0$. Suppose there is a random variable $T$ with a density, and $k \in \text{range}(T)$ such that $\mathbb{P}((A, B) \in E) = \mathbb{P}(T = k)$. Suppose further that for each $a \in \mathcal{A}$, there is a random variable $T_a$, either discrete or with a density, and $k \in \text{range}(T_a)$, such that $\mathbb{P}(B_a = b) = \mathbb{P}(T_a = k)$. Let

$$
\mathcal{L}(A') := \mathcal{L}(A \mid (A, B) \in E), \\
\mathcal{L}(B'_a) := \mathcal{L}(B \mid B_a).
$$

Then, $\mathcal{L}(A', B'_a) = \mathcal{L}((A, B) \mid E)$.

Our recommended approach to sample from $\mathcal{L}(A')$ is rejection sampling. For any measurable function $p : S \to [0, 1]$, let $\mathcal{L}(Z \mid U < p(Z))$ denote the first coordinate of $\mathcal{L}((Z, U) \mid U < p(Z))$, where $U$ is a uniform random variable on $[0, 1]$ independent of $Z$. This allows concise descriptions of distributions resulting from rejection sampling.

**Lemma 6** (DeSalvo 2018) Under the assumptions of Theorem 5 or Theorem 6, pick any finite constant $C \geq \sup_x B(\mathcal{B}_x)$. We have

$$
\mathcal{L}(A') = \mathcal{L} \left( A \mid U < \frac{B(\mathcal{B}_a)}{C} \right).
$$

That is, to sample from $\mathcal{L}(A')$, we first sample from $\mathcal{L}(A)$, and then reject with probability $1 - B(\mathcal{B}_a)/C$.

In our applications of PDC that follow, we indicate the use of PDC by specifying distributions $\mathcal{L}(A)$ and $\mathcal{L}(B)$ and an event $E$ such that the measure of interest is $\mathcal{L}((A, B) \mid (A, B) \in E)$.
4 Main results

4.1 Proof of Theorem 1

The proof follows by PDC and induction. Fix any $1 \leq i \leq m$ and $1 \leq j \leq n$, row sums $r$ and column sums $c$, and define $r(i, k) := (r_1, \ldots, r_{i-1}, r_i - k, r_{i+1}, \ldots, r_m)$ and $c(j, k) := (c_1, \ldots, c_{j-1}, c_j - k, c_{j+1}, \ldots, c_n)$. Then we have

$$\Sigma(r, c, O_{i-1,j}) = \Sigma(r(i, 0), c(j, 0), O_{i,j}) + \Sigma(r(i, 1), c(j, 1), O_{i,j}), \quad 1 \leq i \leq m, 1 \leq j \leq n. \quad (9)$$

Note that $E_{r,c} = E_{r,c}(O_{0,1})$. Let $\epsilon_1$ denote the least significant bit of the $(1, 1)$th entry of the table. Let $A = \{\epsilon_1\}$, and $B = E_{r,c}(O_{0,1})\backslash\{\epsilon_1\}$. We have

$$\mathbb{P}(A = 0 \mid E_{r,c}(O_{0,1})) = \frac{\Sigma(r(1, 0), c(1, 0), O_{1,1}) + \Sigma(r(1, 1), c(1, 1), O_{1,1})}{\Sigma(r(1, 0), c(1, 0), O_{1,1})} = f(1, 1, 0, r, c),$$

and similarly, we have $\mathbb{P}(A = 1 \mid E_{r,c}(O_{0,1})) = f(1, 1, 1, r, c)$. This is the first stage of sampling using PDC. The second stage is to sample from $\mathcal{L}(B\mid E_{r,c}(O_{0,1}), \epsilon_1)$, which is simply the uniform distribution over the set $E_{r(1,\epsilon_1),c(1,\epsilon_1)}(O_{1,1})$.

Denote by $\epsilon_{i,j}$ the least significant bit of the $(i, j)$th entry of the table. Assume the least significant bit of each entry in the first $j - 1$ columns and the first $i - 1$ rows in the $j$th column has already been sampled according to the correct conditional distribution; we denote this set by $\epsilon^{(i,j)} := \{\epsilon_{i,1}, \ldots, \epsilon_{i-1,j}\}$. The task is then to sample from the set $E_{r,c}(O_{0,1})\backslash\{\epsilon_{1,1}, \ldots, \epsilon_{i-1,j}\} = E_{r',c'}(O_{i-1,j})$, where $r' = (r_1', \ldots, r_m')$ satisfies $r_i' = r_i - \sum \epsilon_{i,\ell}$ for all $i$, and similarly $c' = (c_1', \ldots, c_n')$ satisfies $c_j' = c_j - \sum \epsilon_{j,\ell}$ for all $j$, where the sums run over only those elements in $\epsilon^{(i,j)}$. Let $A = \{\epsilon_{i,j}\}$, and $B = E_{r',c'}(O_{i-1,j})\backslash\{\epsilon_{i,j}\}$. Using Eq. (9) and PDC, we sample in the first stage $A$ according to the distribution $\mathbb{P}(A = k \mid E_{r,c}(O_{i-1,j})) = f(i, j, k, r, c)$ for $k \in \{0, 1\}$, and the second stage is again the uniform distribution over the remaining set $B$. We have just shown that the inner two loops over entries sample bits from the appropriate conditional distribution.

After $\epsilon_{m,n}$ has been sampled from the appropriate conditional distribution, the result is a sample from the distribution $\mathcal{L}(\epsilon^{(m,n)}\mid E_{r,c})$, which we denote by $\epsilon_0$, and the remaining distribution to be sampled from is the uniform distribution over the set $E_{r',c'}(O_{m,n})$. However, since all elements of each table in this set are even, and all coordinates of the line sums $r'$ and $c'$ are even, it is easy to see that there is a one-to-one correspondence between each element in $E_{r',c'}(O_{m,n})$ and each element in $E_{r'/2,c'/2}(O_{0,1})$ by a factor of 2 in each corresponding entry.

The second iteration of the outer-most loop of the algorithm generates an element from the distribution $\mathcal{L}(\epsilon^{(m,n)} \mid E_{r'/2,c'/2})$, which we denote by $\epsilon_1$. By Theorem 5, the expression for $i$ in Line 33 is distributed as $\mathcal{L}(\epsilon_0 + 2\epsilon_1 \mid E_{r,c})$. Assume that after the first $b - 1$ iterations of the algorithm, the expression for $i$ in Line 33 is distributed as $\mathcal{L}(\epsilon_0 + 2\epsilon_1 + \cdots + 2^{b-1}\epsilon_{b-1} \mid E_{r,c})$, and that $r'$ is the vector of remaining row sums, and

\( \Box \) Springer
c’ is the vector of remaining column sums given the first b − 1 bits of the entries of the table. Then, the b-th iteration of the algorithm generates \( \mathcal{L}(\epsilon^{(m,n)} | E_r,c) \) which is distributed as \( \mathcal{L}(\epsilon_{0}, \ldots, \epsilon_{b-1}) \). By Theorem 5, the expression for \( t \) in Line 33 is thus distributed as \( \mathcal{L}(\epsilon_{0} + 2\epsilon_{1} + \ldots + 2^{b} \epsilon_{b} | E_r,c) \).

After at most \( \lceil \log_2(M) \rceil + 1 \) iterations, where \( M \) is the largest row sum or column sum, all row sums and column sums will be zero, at which point the algorithm terminates and returns the current table \( t \).

### 4.2 Proof of Theorem 2

In order to decide Line 11, we must compute exactly \( r \) bits of \( f \) with probability \( 2^{-r-1} \). Define function \( c(r, m, n, M; \alpha, \beta, \gamma) := 2^{r} m^\alpha n^\beta \log^\gamma(M) \), and note that

\[
\sum_{r=1}^{\infty} 2^{-r-1} c(r, m, n, M; \alpha, \beta, \gamma) = \frac{1}{2} n^\alpha m^\beta \log^\gamma(M).
\]  

(10)

Assuming Conjecture 1, the cost to evaluate \( f \) to the \( r \)-th bit is \( O(2^{r} m^\alpha n^\beta \log^\gamma(M)) \); therefore, by Eq. (10) above, the cost to decide Line 11 is \( O(m^\alpha n^\beta \log^\gamma(M)) \). Summing over all \( mn \) entries at most \( 1 + \lceil \log_2(M) \rceil \) times gives a total runtime which is \( O(m^{\alpha+1} n^{\beta+1} \log^{\gamma+1}(M)) \), i.e., polynomial in the size of the table and logarithmic in the largest line sum.

### 4.3 Quasi-geometric random variable

In Algorithm 1, after the least significant bit of each entry of the table is sampled, we return to the same sampling problem with reduced row sums and column sums. More importantly, after each step, we choose new parameters \( q_j, j = 1, 2, \ldots, n \), based on these new column sums, which tilts the distribution more in favor of this new table. If we had sampled the geometric random variables directly, it would be equivalent to choosing one value of \( q \) for each entry and using it for the entire algorithm, i.e., sampling from Bernoulli random variables with parameters \( q/(1 + q) \), \( q^2/(1 + q^2) \), \( q^4/(1 + q^4) \), etc. Using this new approach, we are sampling from Bernoulli random variables with parameters \( q/(1 + q) \), \( q’/(1 + q’) \), \( q”/(1 + q”) \), etc., where \( q’, q”, \) etc., are chosen after each iteration, and more effectively target the current set of row sums and column sums.

**Remark 5** One might define a new type of random variable, say, a quasi-geometric random variable, denoted by \( Q \), which is defined as

\[
Q(q, q’, q”, \ldots) = \text{Bern} \left( \frac{q}{1 + q} \right) + 2 \text{Bern} \left( \frac{q’}{1 + q'} \right) + 4 \text{Bern} \left( \frac{q”}{1 + q”} \right) + \cdots,
\]

where all of the Bernoulli random variables are mutually independent. In our case, we choose \( q’ \) after an iteration of the algorithm completes with \( q \). Subsequently,
we use $q''$ after an iteration of the algorithm completes with $q'$, etc. This quasi-
geometric random variable has more degrees of freedom than the usual geometric random variable, although we certainly lose other nice properties which are unique to geometric random variables.

**Remark 6** In Barvinok (2010), a question is posed to obtain the asymptotic joint distribution of a small subset of entries in a random contingency table. We surmise it may be fruitful to consider the joint distribution of the $r$ least significant bits of the usual geometric distribution; or, alternatively, to consider a quasi-geometric random variable defined in Remark 5 with given parameters $q$, $q'$, $q''$, etc.

### 4.4 A probabilistic interpretation of $f$

An equivalent formulation of the function $f$ in Algorithm 1 is given by a joint distribution of random variables, which we now describe.

| Geo($q$) | Geometric distribution with probability of success $1 - q$, for $0 < q < 1$, with $P(\text{Geo}(q) = k) = (1 - q)q^k$, $k = 0, 1, 2, \ldots$. |
|----------|----------------------------------------------------------------------------------------------------------------------------------|
| NB($m$, $q$) | Negative binomial distribution with parameters $m$ and $1 - q$, given by the sum of $m$ independent Geo($q$) random variables, with $P(\text{NB}(m, q) = k) = \binom{m+k-1}{k}(1-q)^m q^k$. |
| $U$ | Uniform distribution on $[0, 1]$. We will also use $U$ in the context of a random variable; $U$ should be considered independent of all other random variables, including other instances of $U$. |
| Bern($p$) | Bernoulli distribution with probability of success $p$. Similarly to $U$, we will also use it as a random variable. |
| $\xi_{i,j}(q)$ | Geo($q$) random variables which are independent for distinct pairs $(i, j)$, $1 \leq i \leq m$, $1 \leq j \leq n$. |
| $\xi'_{i,j}(q, c_j)$ | Random variables which have distribution $L \left( \xi_{i,j}(q) \sum_{\ell=1}^{m} \xi_{\ell,j}(q) = c_j \right)$, and are independent of all other random variables $\xi_{i,\ell}(q)$ for $\ell \neq j$. |
| $2\xi''_{i,j}(q, c_j)$ | Random variables which have distribution $L \left( 2\xi_{i,j}(q^2) \sum_{\ell=1}^{m} 2\xi_{\ell,j}(q^2) = c_j \right)$, and are independent of all other random variables $\xi_{i,\ell}(q)$ for $\ell \neq j$. |
| $\eta'_{i,j,s}(q, c_j)$ | Random variables which have distribution $L \left( \xi_{i,j}(q) \sum_{\ell=1}^{m} 2\xi_{\ell,j}(q^2) + \sum_{\ell=s+1}^{m} \xi_{\ell,j}(q) = c_j \right)$, and are independent of all other random variables $\xi_{i,\ell}(q)$ for $\ell \neq j$. |
| $2\eta''_{i,j,s}(q, c_j)$ | Random variables which have distribution $L \left( 2\xi_{i,j}(q^2) \sum_{\ell=1}^{m} 2\xi_{\ell,j}(q^2) + \sum_{\ell=s+1}^{m} \xi_{\ell,j}(q) = c_j \right)$, and are independent of all other random variables $\xi_{i,\ell}(q)$ for $\ell \neq j$. |
| $q$ | The vector $(q_1, \ldots, q_n)$, where $0 < q_i < 1$ for all $i = 1, \ldots, n$. |
| $R_i$ | $= (\xi_{i,1}, \xi_{i,2}, \ldots, \xi_{i,n})$ for $i = 1, 2, \ldots, m$. |
| $C_j$ | $= (\xi_{1,j}, \xi_{2,j}, \ldots, \xi_{m,j})$ for $j = 1, 2, \ldots, n$. |
We then have
\[
\frac{\mathbb{P}(\epsilon_{1,1} = k|E)}{\mathbb{P}(\epsilon_{1,1} = k)} \propto \mathbb{P}(E|\epsilon_{1,1} = k)
\]
\[
\propto \mathbb{P}\left(2\xi_{1,1}(q_1^2) + \sum_{i=2}^{m} \xi_{i,1}(q_1) = c_1 - k, \quad 2\xi_{1,1}(q_1^2) + \sum_{j=2}^{n} \xi_{1,j}(q_j) = r_1 - k, \quad C_2 = c_2, \quad R_2 = r_2, \quad C_n = c_n, \quad R_m = r_m\right)
\]
\[
\times \mathbb{P}\left(2\eta'_{1,1}(q_1, c_1) + \sum_{j=2}^{n} \eta'_{1,j}(q_1, c_1) = r_1 - k, \quad C_1 = c_1, \quad \epsilon_{1,1} = k, \quad C_2 = c_2, \quad C_n = c_n, \quad R_m = r_m\right)
\]
\[
\times \mathbb{P}\left(2\xi_{1,1}(q_1^2) + \sum_{i=2}^{m} \xi_{i,1}(q_1) = c_1 - k\right).
\]

The right-hand side of the last expression multiplied by \(\mathbb{P}(\epsilon_{1,1} = k)\) is precisely the function \(f(1, 1, k, r, c)\). That is, for all integer-valued arguments \(k \in \{0, 1\}\), and nonnegative vectors \(r = (r_1, \ldots, r_m) \geq 0\), \(c = (c_1, \ldots, c_n) \geq 0\), we have just shown that

\[
f(1, 1, k, r, c) \propto \mathbb{P}\left(\eta'_{1,1}(q_1, c_1) + \sum_{\ell=2}^{n} \xi'_{1,\ell}(q_\ell, c_\ell) = r_1 - k, \quad \eta''_{1,1}(q_1, c_1) + \sum_{\ell=2}^{n} \xi''_{1,\ell}(q_\ell, c_\ell) = r_2, \quad \vdots, \quad \eta''_{m,1,1}(q_1, c_1) + \sum_{\ell=2}^{n} \xi''_{m,\ell}(q_\ell, c_\ell) = r_m\right)
\]
\[
\times \mathbb{P}\left(2\xi_{1,1}(q_1^2) + \sum_{i=2}^{m} \xi_{i,1}(q_i) = c_1 - k\right).
\]

(11)

Recall that Algorithm 1 samples \(\mathcal{L}(\epsilon_{1,1}|E), \mathcal{L}(\epsilon_{2,1}|E, \epsilon_{1,1}), \mathcal{L}(\epsilon_{3,1}|E, \epsilon_{1,1}, \epsilon_{2,1}),\) etc., until the entire first column is sampled. Then it starts with the top entry of the
second column and samples the least significant bits from top to bottom according to the conditional distribution. Generalizing Eq. (11), for all integer-valued arguments \(1 \leq i \leq m, 1 \leq j \leq n, k \in \{0,1\}\), and nonnegative vectors \(r = (r_1, \ldots, r_m) \geq 0, c = (c_1, \ldots, c_n) \geq 0\), we have

\[
f(i, j, k, r, c) \propto P\left( \sum_{\ell=1}^{j-1} 2^{\ell} \xi_{i, \ell} (q_{j}^2, c_{\ell}) + \sum_{\ell=1}^{j} \eta_{i,j,i} (q_j, c_j) + \sum_{\ell=1}^{j} \xi_{2,j,\ell} (q_{\ell}, c_{\ell}) = r_1 \right) \\
\sum_{\ell=1}^{j-1} 2^{\ell} \xi_{i-1, \ell} (q_{j}^2, c_{\ell}) + \sum_{\ell=1}^{j} \eta_{i-1,j,i} (q_j, c_j) + \sum_{\ell=1}^{j} \xi_{2,j,\ell} (q_{\ell}, c_{\ell}) = r_{i-1} \\
\sum_{\ell=1}^{j-1} 2^{\ell} \xi_{i, \ell} (q_{j}^2, c_{\ell}) + \sum_{\ell=1}^{j} \eta_{i,j,i} (q_j, c_j) + \sum_{\ell=1}^{j} \xi_{2,j,\ell} (q_{\ell}, c_{\ell}) = r_i - k \\
\sum_{\ell=1}^{j-1} 2^{\ell} \xi_{i+1, \ell} (q_{j}^2, c_{\ell}) + \sum_{\ell=1}^{j} \eta_{i+1,j,i} (q_j, c_j) + \sum_{\ell=1}^{j} \xi_{2,j,\ell} (q_{\ell}, c_{\ell}) = r_{i+1} \\
\vdots \\
\sum_{\ell=1}^{j-1} 2^{\ell} \xi_{m, \ell} (q_{j}^2, c_{\ell}) + \sum_{\ell=1}^{j} \eta_{m,j,i} (q_j, c_j) + \sum_{\ell=1}^{j} \xi_{2,j,\ell} (q_{\ell}, c_{\ell}) = r_m \\
\times P\left( \sum_{\ell=1}^{i} 2 \xi_{\ell,j} (q_{j}^2) + \sum_{\ell=i+1}^{m} \xi_{\ell,j} (q_{\ell}) = c_j - k \right) \\
\times P (\epsilon_{i,j} = k). \quad (12)
\]

Note that \(\sum_{\ell=1}^{i} \xi_{\ell,j} (q_{j}^2)\) is the sum of \(i\) i.i.d. geometric random variables, and is hence a negative binomial distribution with parameters \(i\) and \(q_{j}^2\). We thus have a linear combination of two independent negative binomial distributions:

\[
P\left( \sum_{\ell=1}^{i} 2 \xi_{\ell,j} (q_{j}^2) + \sum_{\ell=i+1}^{m} \xi_{\ell,j} (q_{\ell}) = c_j - k \right) = P\left( 2 \text{NB}(i, q_{j}^2) + \text{NB}(m-i, q_{j}) = c_j - k \right).
\]

We note that each random variable in the probability above has an explicitly computable and simple probability mass function, which we write below.

\[
P\left( \xi_{i,j} (q, c_j) = k \right) = P\left( \xi_{i,j} (q) = k \right) \frac{\text{NB}(m-1, q)(c_j-k)}{\text{NB}(m, q)(c_j)} \\
= \left( \frac{(m-1)+(c_j-k)-1}{c_j-k} \right)^{c_j-k} \cdot \frac{1}{(m+c_j-1)c_j}, \quad k = 0, 1, \ldots, c_j. \quad (13)
\]

Denote by \(\epsilon_{i,j}\) the observed value of the parity bit of \(\xi_{i,j}\), \(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\). Let \(c_j' := \frac{c_j-\sum_{i=1}^{m} \epsilon_{i,j}}{2}\). We have

\(\copyright\) Springer
\[
\Pr \left( 2\xi_{i,j} (q, c_j') = k \right) = \Pr \left( 2\xi_{i,j} (q^2) = k \right) \frac{\text{NB}(m - 1, q^2) \left\{ c'_j - \frac{k}{2} \right\}}{\text{NB}(m, q^2) \left\{ c'_j \right\}} = \frac{\left( \frac{(m - 1) + \left( c'_j - \frac{k}{2} \right) - 1}{c'_j - \frac{k}{2}} \right)}{\left( \frac{m + c'_j - 1}{c'_j} \right)}, \quad k = 0, 2, 4, \ldots, 2c'_j. \tag{14}
\]

Let \( c''_j := c_j - \sum_{\ell=1}^s \epsilon_{\ell,j}, \ j = 1, 2, \ldots, n \). We have

\[
\Pr \left( 2\eta_{i,j,s} (q, c''_j) = k \right) = \Pr \left( 2\xi_{i,j} (q^2) = k \right) \frac{\Pr \left( 2\text{NB}(s - 1, q^2) + \text{NB}(m - s, q) = c''_j - k \right)}{\Pr \left( 2\text{NB}(s, q^2) + \text{NB}(m - s, q) = c''_j \right)}, \tag{15}
\]

for \( k = 0, 2, \ldots, 2 \left\lfloor \frac{c''_j}{2} \right\rfloor \), and

\[
\Pr \left( \eta_{i,j,s} (q, c''_j) = k \right) = \Pr \left( \xi_{i,j} (q) = k \right) \frac{\Pr \left( 2\text{NB}(s, q^2) + \text{NB}(m - s - 1, q) = c''_j - k \right)}{\Pr \left( 2\text{NB}(s, q^2) + \text{NB}(m - s, q) = c''_j \right)}, \tag{16}
\]

for \( k = 0, 1, \ldots, c''_j \).

### 4.5 Approximate sampling of integer-valued tables

Section 4.4 demonstrates that the least significant bit of entry \((i, j)\) is plausibly close in distribution to \( \mathcal{L} \left( \text{Bern} \left( \frac{q_j}{1+q_j} \right) \right) \), with \( q_j = \frac{c_j}{m+c_j} \). An approach which shares many similar characteristics of the Boltzmann sampler is to sample the least significant bits of the table one at a time according to this unconditional distribution, without applying any other rejections other than the parity constraints, and then apply the recursion until all row sums and column sums are 0.

We suggest a slightly more involved approach, which is to treat the row sum conditions as essentially independent, and reject each bit generated as if it was independent of all other rows. Algorithm 3 below uses Eq. (12) and assumes that the rows are independent, which greatly simplifies the rejection probability formula. A rejection function is then given by
\[ F(i, j, m, n, q, r, c, k, \epsilon) \]

\[
\begin{align*}
F(i, j, m, n, q, r, c, k, \epsilon) := &\ P\left(\sum_{\ell=1}^{j-1} 2\xi_{i,\ell}^\prime(q_{\ell}^2, c_j) + 2\eta_{i,j,i}^\prime + \sum_{\ell=j+1}^{n} \xi_{i,\ell}^\prime(q_{\ell}, c_j) = r_i - k\right) \\
\times &\ P\left(\sum_{\ell=1}^{i} 2\xi_{i,j,\ell}(q_j^2) + \sum_{\ell=i+1}^{m} \xi_{\ell,j}(q_j) = c_j - k\right), \quad k \in \{0, 1\}. \quad (17)
\end{align*}
\]

Note that the first term is a probability over a sum of independent random variables, and as such can be computed using convolutions in time \(O(M^2)\) or fast Fourier transforms in time \(O(n M \log M)\). Further speedups are possible since only the first few bits of the function are needed on average.

**Algorithm 3** Generation of approximately-uniformly random \((r, c)\)-contingency table

1: \(M \leftarrow \max(\max_i r_i, \max_j c_j)\).
2: \(t \leftarrow m \times n\) table with all 0 entries.
3: for \(b = 0, 1, \ldots, \lceil \log_2(M) \rceil \) do
4: \(\sigma = \text{any permutation such that } \sigma \circ r \text{ is in increasing order.}\)
5: \(\sigma = \text{any permutation such that } \sigma \circ c \text{ is in increasing order.}\)
6: \(r \leftarrow \sigma \circ r.\)
7: \(c \leftarrow \sigma \circ c.\)
8: for \(j = 1, \ldots, n - 1\) do
9: \(\text{for } i = 1, \ldots, m - 1 \text{ do}\)
10: \(q_j \leftarrow c_j/(m + c_j).\)
11: \(\epsilon_{i,j} \leftarrow \text{Bern}(q_j/(1 + q_j)).\)
12: if \(U > \max_{\ell \in \{0, 1\}} F(i, j, m, n, q, r, c, \epsilon)\) then
13: \(\text{goto Line 11.}\)
14: \(\text{end if}\)
15: \(r_i \leftarrow r_i - \epsilon_{i,j}.\)
16: \(c_j \leftarrow c_j - \epsilon_{i,j}.\)
17: \(\text{end for}\)
18: \(c_{m,j} \leftarrow c_j \mod 2\)
19: \(c_j \leftarrow c_j - \epsilon_{m,j}.\)
20: \(c_j \leftarrow c_j/2\)
21: \(r_m \leftarrow r_m - \epsilon_{m,j}.\)
22: \(\text{end for}\)
23: for \(i = 1, \ldots, m\) do
24: \(\epsilon_{i,n} \leftarrow r_i \mod 2\)
25: \(r_i \leftarrow r_i - \epsilon_{i,n}\)
26: \(r_i \leftarrow r_i/2\)
27: \(c_n \leftarrow c_n - \epsilon_{i,n}\)
28: \(\text{end for}\)
29: \(c_n \leftarrow c_n/2.\)
30: \(\epsilon \leftarrow \sigma^{-1}_R \circ (\text{Apply permutation to rows})\)
31: \(\epsilon \leftarrow \sigma^{-1}_C \circ (\text{Apply permutation to columns})\)
32: \(t \leftarrow t + 2^b \epsilon.\)
33: \(\text{end for}\)
4.6 Proof of Theorem 3

Since we are normalizing by the max over the two states, at least one of the states is accepted with probability $1$, and so the total number of rejections is bounded from above by the wait time until this state is generated. Since the random variable generating the bit is Bernoulli with parameter $\frac{q_j}{1+q_j}$, we have

\[
P\left(\text{Bern}\left(\frac{q_j}{1+q_j}\right) = 0\right) = \frac{m + c_j}{m + 2c_j} \geq \frac{1}{2}.
\]

\[
P\left(\text{Bern}\left(\frac{q_j}{1+q_j}\right) = 1\right) = \frac{c_j}{m + 2c_j} \geq \frac{1}{m + 2}.
\]

Thus, at worst we accept a bit with proportion $m/2$ times more likely than the other state. Each entry is therefore rejected at most an expected $O(m)$ number of times. The rejection function needs to be computed at worst $O(m M \log M)$ times, with a cost of $O(n M \log M)$ for performing an $n$-fold convolution via fast Fourier transforms.

4.7 $2 \times n$ tables

In this section we consider $2 \times n$ tables only, and assume WLOG that $r_1 \geq r_2$ and also $c_1 \geq c_2 \geq \cdots \geq c_n$. Let $N = r_1 + r_2$ denote the sum of all entries in the table, and $M = \max(r_1, c_1)$.

Dyer and Greenhill (2000) described a $O(n^2 \log N)$ asymptotically uniform MCMC algorithm based on updating a $2 \times 2$ submatrix at each step. Kijima and Matsui (2006) adapted the same chain using coupling from the past to obtain an exactly uniform sampling algorithm at the cost of an increased run time of $O(n^3 \log N)$.

**Proof of Theorem 4** Let $\xi$ be a $2 \times n$ $(r, c)$-contingency table, and let $\xi' = (\xi_{12}, \xi_{13}, \ldots, \xi_{1,n})$; that is, the top row without the first entry. Since $r$ and $c$ are fixed, there is a bijective relationship between $\xi$ and $\xi'$; each determines the other. Then,

\[
P[ x = \xi ] = \frac{1}{(c_2 + 1)(c_3 + 1) \cdots (c_n + 1)}.
\]

This does not depend on $\xi$, so $x$ is uniform restricted to $(r, c)$-contingency tabs.

By Theorem 7.1 of DeSalvo (2018) row of entries $x_{12}, \ldots, x_{1,n}$ is accepted if and only if $x_{11} = r_1 - x_{12} - \cdots - x_{1,n-1}$ lies between 0 and $c_1$, which occurs with probability $p_{r,c} = \mathbb{P}(U_1 + \cdots + U_{n-1} \in [\max(0, r_1 - c_1), r_1])$.

The number of times we must repeat the random sampling of these $n-1$ uniform random variables is geometrically distributed with expected value $1/p_{r,c}$. Letting $t_{r,c}$ denote the cost of sampling $n-1$ uniform random variables, the expected runtime cost of the algorithm is then exactly $t_{r,c}/p_{r,c}$. If we assume fixed floating point calculations, then $t_{r,c} = O(n)$, otherwise we have $t_{r,c} = O(n \log(c_1))$, for all possible row sums $r = (r_1, r_2)$ and column sums $c = (c_1, \ldots, c_n)$, and hence the result follows. \[\square\]
Proof of Corollary 1 When all column sums are equal, say to $c \geq 1$, we have $U_i$ is uniformly distributed over the interval $\{0, \ldots, c\}$ for $i = 2, 3, \ldots, n$. Let $U := U_2 + \cdots + U_n$, then we have $\mathbb{E} U = \frac{c}{2} (n - 1)$ and $\text{Var}(U) = (n - 1) \frac{(c+1)^2 - 1}{12}$. We have $r_1 + r_2 = n c$, and so when all row sums are equal, we have $r \equiv r_1 = r_2 = n c/2$. Define $a(c) := \frac{c}{\sqrt{(c+1)^2 - 1}/12}$, and $z_n := a(c)/\sqrt{n - 1}$. Letting $Z$ denote a standard normal random variable, by the central limit theorem we have

$$p_{r,c} = \mathbb{P}(U \in [r - c, r]) \approx \mathbb{P}(Z \in [-z_n, z_n]) = O(\sqrt{n}).$$

Note in particular that the big-O expression is also independent of the value of $c$, since $a(c)$ is bounded both from above and below by fixed constants for all $c \geq 1$. \qed

There is a key observation at this point, which is that exponential random variables share the same property that $\xi_1, j$ given $\xi_1, j + \xi_2, j = c_j$ is uniform over the continuous interval $[0, c_j]$. The algorithm for real-valued $2 \times n$ tables is presented below in Algorithm 4, and is essentially the same as Algorithm 2.

**Algorithm 4** Generating a uniformly random $2 \times n$ real-valued $(r, c)$-contingency table.

1: for $j = 2, \ldots, n$ do
2: choose $x_{1,j}$ uniformly from $[0, c_j]$ 
3: let $x_{2,j} = c_j - x_{1,j}$ 
4: end for
5: let $x_{1,1} = r_1 - \sum_{j=2}^n x_{1,j}$ 
6: let $x_{2,1} = r_2 - \sum_{j=2}^n x_{2,j}$ 
7: if $x_{1,1} < 0$ or $x_{2,1} < 0$ then
8: restart from Line 1 
9: end if 
10: return $x$

**Theorem 7** Algorithm 4 produces an exact, uniformly random $2 \times n$ real-valued $(r, c)$-contingency table, with a total runtime cost which is $O(n \log(c_1)/p_{r,c})$, where $p_{r,c}$ is defined as

$$p_{r,c} := \mathbb{P}(U_2 + \cdots + U_n \in [\max(0, r_1 - c_1), r_1]),$$

where $U_2, U_3, \ldots, U_n$ denote independent uniform random variables, with $U_j$ uniform over the continuous interval $[0, c_j]$, $j = 2, \ldots, n$.

**Corollary 2** Assuming all column sums are equal, and all row sums are equal, the total runtime cost of Algorithm 4 is $O(n^{3/2} \log(c_1))$.

We observe that the algorithm also runs quickly when $r_1 \approx \mathbb{E}[U_1+\cdots+U_n] = N/2$, i.e., the two row sums are similar in size (but not necessarily equal), and also when $c_1$ is large. It follows that having a skewed distribution of column sums and an even distribution of row sums is advantageous to runtime, which we now characterize.
Table 1  Simulated runtime under Algorithm 4 for sampling contingency tables with homogeneous row and column sums, compared to optimized rejection sampling where columns are picked using a discrete uniform random variable

| Rows | Columns | Density | Rejections | Runtime  |
|------|---------|---------|------------|----------|
| 2    | 2       | 5       | 0*         | 269 ns*  |
| 2    | 10      | 5       | 1.32*      | 1.13 μs* |
| 2    | 100     | 5       | 6.22*      | 23.0 μs* |
| 2    | 1000    | 5       | 21.6†      | 665 µs†  |
| 2    | 10⁴     | 5       | 69.3†      | 19.7 ms† |
| 2    | 10⁵     | 5       | 199†       | 580 ms†  |
| 2    | 10⁶     | 5       | 755‡       | 23.7 s‡  |

The symbols *, † and ‡ denote averages over a sample of size $10^6$, 1000 and 1 respectively. As predicted analytically, the number of rejections grows as $O(\sqrt{n})$ while the runtime grows as $O(n^{3/2} \log N)$.

Corollary 3  Suppose $U_2 + \cdots + U_n$ satisfies the central limit theorem. Assume there exists some $t \in \mathbb{R}$ such that, as $n \to \infty$, we have

$$\frac{r_1 - c_1 - \frac{c_2 + \cdots + c_n}{2}}{\sqrt{\frac{c_2^2 + \cdots + c_n^2}{12}}} \to t.$$

Then, asymptotically as $n \to \infty$, the runtime cost of Algorithm 2 is $O(n \left(\Phi(t + c'_1) - \Phi(t)\right))$.

Proof  Letting $Z$ denote a standard normal random variable, under our assumptions we have

$$\mathbb{P}(U_2 + \cdots + U_n \in [r_1 - c_1, r_1]) \sim \mathbb{P}(Z \in [t, t + c'_1]).$$

Corollary 4  Suppose $U_2 + \cdots + U_n$ satisfies the central limit theorem. Suppose $c'_1 \to \lambda \in (0, \infty]$. Then the runtime cost of Algorithm 2 is $O(n)$.

Corollary 4 says that when the square of the largest column sum dominates the sum of squares of the remaining column sums, then the majority of the uncertainty is in column 1, which is handled optimally by PDC.

The following Table 1 demonstrates that under the conditions of Corollary 1, i.e., equal row sums and column sums, the expected number of rejections grows like $O(\sqrt{n})$, and the expected runtime grows like $O(n^{3/2} \log(c_1))$.  

© Springer
It is also possible to adapt the approach in Barvinok and Hartigan (2012), that of optimizing the joint distribution of independent geometric random variables, i.e., the computing of the maximum entropy matrix, for use in PDC. We start with the observation that, for two independent geometric random variables $X_1(q_1)$ and $X_2(q_2)$, where $0 < q_1, q_2 < 1$ denote the failure probabilities of two independent Bernoulli trials, we have

$$
P(X(q_1) = j | X_1(q_1) + X_2(q_2) = k) = \left( \frac{q_1}{q_2} \right)^j \left( \sum_{\ell=0}^{k} \left( \frac{q_1}{q_2} \right)^\ell \right)^{-1}, \quad j = 0, 1, \ldots, k. \quad (19)
$$

Note that when $q_1 = q_2$, i.e., in the i.i.d. case, this is the uniform distribution over the set $\{0, 1, \ldots, k\}$, which we utilized above. Define random variable

$$X'(q_1, q_2, k) \overset{D}{=} (X_1(q_1) | X_1(q_1) + X_2(q_2) = k),$$

with probability mass function $g(j; q_1, q_2, k)$ given by Eq. (19) above. Finally, define

$$g^*(q_1, q_2, k) = \max_j g(j; q_1, q_2, k).$$

**Theorem 8** Algorithm 5 uniformly samples from $2 \times n$ contingency tables.

The proof is a routine application of PDC deterministic second half, and is left to the reader.

**Algorithm 5** Algorithm to generate a uniformly random $2 \times n$ $(r, c)$-contingency table via the maximum entropy matrix and PDC.

1: $(s_{1,1}, \ldots, s_{2,n}) \leftarrow$ maximum entropy matrix given $(r, c)$.
2: for $j = 2, \ldots, n$ do
3: \quad $q_{1,j} \leftarrow \frac{s_{1,j}}{1+s_{1,j}}$
4: \quad $q_{2,j} \leftarrow \frac{s_{2,j}}{1+s_{2,j}}$
5: \quad choose $x_{1,j}$ according to the distribution $X'(q_1, j, q_2, c_j)$ given in (19).
6: \quad let $x_{2,j} = c_j - x_{1,j}$
7: end for
8: $x_{1,1} \leftarrow r_1 = \sum_{j=2}^{n} x_{1,j}$
9: $x_{2,1} \leftarrow r_2 = \sum_{j=2}^{n} x_{2,j}$
10: $q_{1,1} \leftarrow \frac{x_{1,1}}{1+x_{1,1}}$
11: $q_{2,1} \leftarrow \frac{x_{2,1}}{1+x_{2,1}}$
12: if $U \geq g(x_{1,1}; q_1, 1, q_1, 2, c_1)$ then
13: \quad restart from Line 1
14: end if
15: return $x$
We end this section by noting that this same approach generalizes in a natural manner to $2 \times 2 \times \cdots \times n$ contingency tables, for which there is also considerable interest; see for example Kieffer et al. (2012), Matsui et al. (2004) and the references therein.

### 4.8 Approximate sampling of real-valued tables

In addition to sampling from nonnegative integer-valued tables, one may also wish to sample from nonnegative real-valued tables with real-valued row sums and column sums. For a real-valued contingency table, our approach also applies, with the role of geometric random variables replaced by exponential random variables. Instead of sampling from the smallest bit of each entry, we instead sample the fractional parts first, and what remains is an integer-valued table with integer-valued row sums and column sums.

To obtain a good candidate distribution for the fractional part of a given entry, we utilize two well-known facts summarized in the lemma below. For real $x$, \{x\} denotes the fractional part of $x$, and \lfloor x \rfloor$ denotes the integer part of $x$, so that $x = \lfloor x \rfloor + \{x\}$. 

**Lemma 7** Let $Y$ be an exponentially distributed random variable with parameter $\lambda > 0$, then:

- the integer part, \lfloor Y \rfloor, and the fractional part, \{Y\}, are independent (Steutel and Thiemann 1987; Von Neumann 1951);
- \lfloor Y \rfloor is geometrically distributed with parameter $1 - e^{-\lambda}$, and \{Y\} has density $f_{\lambda}(x) = \lambda e^{-\lambda x}/(1 - e^{-\lambda})$, $0 \leq x < 1$.

In this case, the recommended approach is the following:

1. Sample the fractional part of the entries of the table first, in proportion to the number of possible integer-valued tables which satisfy the remaining constraints;
2. Sample the remaining integer part of the table.

If an exact sample of each of the items above can be obtained, then an application of PDC implies that the sum of the entries of the two tables has the uniform distribution over nonnegative real-valued tables.

There already exists a polynomial time algorithm for the random sampling of $(r, c)$-contingency tables with real-valued entries contained in Dyer et al. (1991), based on convex polytope sampling, which offers an arguably distinct approach to Markov chains. This section is presented to demonstrate the robustness of the PDC approach, and also highlight a property of exponential random variables which makes them particularly amenable to a probabilistic divide-and-conquer approach.

By Lemma 4, a uniformly random $(r, c)$-contingency table with real-valued entries has distribution $L(\xi \mid G_{r,c})$, where $\xi = (\xi_{ij})$, $1 \leq i \leq m, 1 \leq j \leq n$, is an $m \times n$ matrix of independent exponential random variables with parameter $\lambda_{ij} = -\log(1 - p_{ij})$. An exponential random variable $E(\lambda_{ij})$ can be decomposed into a sum

$$E(\lambda_{ij}) = A(\lambda_{ij}) + G(p_{ij}),$$
where $A(\lambda)$ is a random variable with density
\begin{equation}
    f(x) = \lambda e^{-\lambda x} (1 - e^{-\lambda})^{-1}, \quad 0 \leq x \leq 1,
\end{equation}
and $G(p)$ is a geometric random variable with probability of success at each trial given by $p$, independent of $A(\lambda)$.

One could attempt to adapt the bit-by-bit rejection of Algorithm 1 in the continuous setting to fractional parts, however, instead of counting the number of residual tables, one would have to look at the density of such tables with respect to Lebesgue measure on $\mathbb{R}^{(m-1) \times (n-1)}$. Instead, we present an approximate sampling algorithm analogous to Algorithm 3 which circumvents such calculations.

For each $1 \leq i \leq m$, $1 \leq j \leq n$, let $Y_{i,j}(\lambda_{i,j})$ denote independent exponential random variables with parameter $\lambda_{i,j}$, and let $Y'_{i,j}(\lambda_{i,j})$ denote a random variable with distribution
\begin{equation}
    \mathcal{L}(Y'_{i,j}(\lambda_{i,j})) = \mathcal{L} \left( Y_{i,j}(\lambda_{i,j}) \middle| \sum_{\ell=1}^m Y_{\ell,j}(\lambda_{i,j}) = c_j \right).
\end{equation}

In what follows we will take $\lambda_{i,j} = -\log \left( \frac{c_j}{m+c_j} \right)$, i.e., $\lambda_{i,j}$ does not vary with parameter $i$, and let $q_{i,j} = \frac{c_j}{m+c_j}$. Let $[Y'_{i,j,s}]$ denote a random variable with distribution
\begin{equation}
    \mathcal{L}([Y'_{i,j,s}]) = \mathcal{L} \left( [Y_{i,j}] \middle| \sum_{i=1}^s \xi_{i,j}(q_{i,j}) + \sum_{i=s+1}^m Y_{i,j}(\lambda_{i,j}) = c_j \right).
\end{equation}

A suggested rejection function is then
\begin{equation}
    G(i, j, x, r, c) := \mathbb{P} \left( \sum_{\ell=1}^{i-1} \xi'_{i,\ell}(q_{i,\ell}, c_{i,\ell}) + 2[Y'_{i,j,i}(\lambda_{i,j})] + \sum_{\ell=j+1}^n Y'_{i,\ell}(\lambda_{i,\ell}) \in d(r_i - x) \right)
    \times \mathbb{P} \left( \sum_{\ell=1}^{i} Y_{\ell,j} \middle| \sum_{\ell=i+1}^m Y_{\ell,j} \in d(c_j - x) \right), \quad \text{for } x \in [0, 1].
\end{equation}

In Algorithm 6 below, we let FractionExp$\lambda$ denote the distribution of the fractional part of an exponential random variable with parameter $\lambda$. 
Algorithm 6 Generation of fractional parts of an approximately uniform random \((r, c)\)-real-valued table

1: Let \(\sigma_R\) denote any permutation such that \(\sigma_R \circ r\) is in increasing order.
2: Let \(\sigma_C\) denote any permutation such that \(\sigma_C \circ c\) is in increasing order.
3: \(r \leftarrow \sigma_R \circ r\).
4: \(c \leftarrow \sigma_C \circ c\).
5: for \(j = 1, \ldots, n - 1\) do
6:   for \(i = 1, \ldots, m - 1\) do
7:       \(\lambda_j \leftarrow -\log \left( \frac{c_j}{m + c_j} \right)\).
8:       \(\epsilon_{i,j} \leftarrow \text{FractionExp}(\lambda_j)\).
9:       if \(U > G(i, j, \epsilon_{i,j}, r, c) / \max_{x \in [0,1]} G(i, j, x, r, c)\) then
10:          goto Line 8.
11:       end if
12:       \(r_i \leftarrow r_i - \epsilon_{i,j}\).
13:       \(c_j \leftarrow c_j - \epsilon_{i,j}\).
14:   end for
15: \(\epsilon_{m, j} \leftarrow \{c_j\}\).
16: \(c_j \leftarrow c_j - \epsilon_{m, j}\).
17: \(r_m \leftarrow r_m - \epsilon_{m, j}\).
18: end for
19: for \(i = 1, \ldots, m\) do
20:   \(\epsilon_{i,n} \leftarrow \{r_i\}\).
21: \(r_i \leftarrow r_i - \epsilon_{i,n}\).
22: \(c_n \leftarrow c_n - \epsilon_{i,n}\).
23: end for
24: \(\epsilon \leftarrow \sigma_R^{-1} \circ \epsilon\) (Apply permutation to rows)
25: \(\epsilon \leftarrow \sigma_C^{-1} \circ \epsilon\) (Apply permutation to columns)
26: return \(\epsilon\).

Algorithm 7 Generation of approximately uniform random real-valued \((r, c)\)-table

1: \(\epsilon \leftarrow \) output of Algorithm 6 with \((r, c)\) input.
2: \(r'_i \leftarrow r_i - \sum_{j=1}^{n} \epsilon_{i,j}, i = 1, \ldots, m\).
3: \(c'_j \leftarrow c_j - \sum_{i=1}^{m} \epsilon_{i,j}, j = 1, \ldots, n\).
4: \(t \leftarrow \) output of Algorithm 3 with \((r', c')\) input.
5: Return \(t + \epsilon\).

5 Other tables

One can more generally sample from a table having independent entries with marginal distributions \(\mathcal{L}(X_{1,1})\), \(\mathcal{L}(X_{1,2})\), \ldots, \(\mathcal{L}(X_{m,n})\), i.e.,

\[
\mathcal{L}(X) = \mathcal{L}((X_{i,j})_{1 \leq i \leq m, 1 \leq j \leq n} | E).
\] (20)

If the rejection probabilities can be computed, then we can apply a variation of Algorithm 1, and under mild assumptions also a variation of Algorithm 7.

It is sometimes possible to circumvent the column-rejection probabilities. We now state Algorithm 8, which is a general procedure of independent interest that samples from a conditional distribution of the form
\( \mathcal{L} \left( \left( X_1, X_2, \ldots, X_n \right) \mid \sum_{i=1}^{n} X_i = k \right) \); see Lemma 8 for the explicit form of the rejection probability.

Algorithm 8 Random generation from \( \mathcal{L}((X_1, X_2, \ldots, X_n) \mid \sum_{i=1}^{n} X_i = k) \)

1: Assume:
\( \mathcal{L}(X_1), \mathcal{L}(X_2), \ldots, \mathcal{L}(X_n) \) are independent and either
    all discrete with \( \mathbb{P}(\sum_{i=1}^{n} X_i = k) > 0 \);
    or \( \mathcal{L}(\sum_{i=1}^{n} X_i) \) has a bounded density with \( k \in \text{range}(\sum_{i=1}^{n} X_i) \).
2: if \( n = 1 \) then
3:    return \( k \)
4: end if
5: let \( \nu \) be any value in \( \{1, \ldots, n\} \).
6: for \( i = 1, \ldots, \nu \) do
7:    generate \( X_i \) from \( \mathcal{L}(X_i) \).
8: end for
9: let \( a = (x_1, \ldots, x_\nu) \)
10: let \( s = \sum_{i=1}^{\nu} x_i \).
11: if \( U \geq t(a) \) (See Lemma 8) then
12:    restart
13: else
14:    Recursively call Algorithm 8 on \( \mathcal{L}(X_{\nu+1}), \ldots, \mathcal{L}(X_n) \), with target sum \( k - s \),
    and set \( (X_{\nu+1}, \ldots, X_n) \) equal to the return value.
15: return \( (X_1, \ldots, X_n) \).
16: end if

Lemma 8 Suppose \( X_1, \ldots, X_n \) are independent real-valued random variables. Suppose also that for any \( \nu \in \{1, \ldots, n\} \), and for each \( a = (x_1, \ldots, x_\nu) \in \mathbb{R}^\nu \) and \( k \in \text{range}(\sum_{i=1}^{n} X_i) \), we have either

1. \( \mathcal{L}(\sum_{i=\nu+1}^{n} X_i) \) is discrete,
\[
    t(a) = t(x_1, \ldots, x_\nu) = \frac{\mathbb{P}(\sum_{i=\nu+1}^{n} X_i = k - \sum_{i=1}^{\nu} x_i | (x_1, \ldots, x_\nu))}{\max_{\ell} \mathbb{P}(\sum_{i=\nu+1}^{n} X_i = \ell)} \tag{21}
\]

or

2. \( \mathcal{L}(\sum_{i=\nu+1}^{n} X_i) \) has a bounded density, denoted by \( f_{\nu,n+1} \), and
\[
    t(a) = t(x_1, \ldots, x_\nu) = \frac{f_{\nu,n+1}(k - \sum_{i=1}^{\nu} x_i | (x_1, \ldots, x_\nu))}{\max_{\ell} f_{\nu,n+1}(\ell)} \tag{22}
\]

Then Algorithm 8 samples from \( \mathcal{L}((X_1, X_2, \ldots, X_n) \mid \sum_{i=1}^{n} X_i = k) \).

Proof The rejection probability \( t(a) \) is defined, depending on the setting, by Eq. (21) or Eq. (22), so that once the algorithm passes the rejection step in Line 11, then for any \( 1 \leq \nu \leq n \), the vector \( (X_1, \ldots, X_\nu) \) has distribution \( \mathcal{L}(A|h(A, B) = 1) \), where
\( A = (X_1, \ldots, X_n) \) and \( h(A, B) = \mathbb{1}(\sum_{i=1}^n X_i = k) \). Let \( a \) denote the observed value in this stage.

We now use induction on \( n \). When \( n = 1 \), we take \( A = (X_1) \) and \( B = \emptyset \), then Algorithm 8 with input \((\mathcal{L}(X_1), k)\) returns the value of the input target sum \( k \) for any \( k \in \text{range}(X_1) \), which has distribution \( \mathcal{L}(X_1 | X_1 = k) \).

Assume, for all \( 1 \leq \nu \leq n \), Algorithm 8 with input \((\mathcal{L}(X_{\nu+1}), \ldots, \mathcal{L}(X_n), \ell)\) returns a sample from \( \mathcal{L}(X_{\nu+1}, \ldots, X_n | h(a, B) = \ell) \), for any \( \ell \in \text{range}(\sum_{i=1}^n X_i) \); i.e., it returns a sample from \( \mathcal{L}(B | h(a, B) = 1) \), say \( b \), where \( B = (X_{\nu+1}, \ldots, X_n) \).

Hence, each time Algorithm 8 is called, it first generates a sample from distribution \( \mathcal{L}(A | h(a, B) = 1) \), and then the return value of the recursive call in Line 15 returns a sample from \( \mathcal{L}(B | h(a, B) = 1) \). By Lemma 6, \((a, b)\) is a sample from \( \mathcal{L}((A, B) | h(a, B) = 1) \).

In the case where computing the row rejection probabilities after the generation of each column is not practical, we recommend independent sampling of columns \( 1, \ldots, n - 1 \) all at once, with a single application of PDC deterministic second half for the generation of the final column. This approach is more widely applicable, as it requires very little information about the marginal distribution of each entry.

Let \( X'_{i,n} \) denote the random variable with distribution \( \mathcal{L}(X_{i,n} | \sum_{i=1}^m X_{i,n}) \). In the following algorithm, the function used for the rejection probability, when \( X'_{i,n} \) is discrete, is given by (where we recall the notation that \( r_i \) denotes the row sum of row \( i \), \( i = 1, \ldots, m \))

\[
    h(i, \mathcal{L}(X_{i,n}'), k) = \mathbb{P}(X_{i,n}' = r_i - k),
\]

and when \( X'_{i,n} \) is continuous with bounded density \( f_{X_{i,n}'} \), is given by

\[
    h(i, \mathcal{L}(X_{i,n}'), k) = f_{X_{i,n}'}(r_i - k).
\]

Thus, for Algorithm 9 below, we simply need to be able to compute the distribution \( \mathcal{L}(X_{i,n}') \) and find its maximum point probability, for \( i = 1, 2, \ldots, m \).

**Algorithm 9** Generating a random variate from \( \mathcal{L}(X) \) specified in Eq. (20).

```plaintext
1: for \( j = 1, \ldots, n - 1 \) do
2:   let \( c_j \) denote the return value of Algorithm 8 using input \((\mathcal{L}(X_{1,j}), \ldots, \mathcal{L}(X_{m,j}), c_j)\).
3: end for
4: let \( x_{i,n} = r_i - \sum_{j=1}^{n-1} x_{i,j} \) for \( i = 1, \ldots, m \).
5: if \( U \geq \prod_{i=1}^m \sup_{h(i, \mathcal{L}(X_{i,n}'))} \frac{h(i, \mathcal{L}(X_{i,n}'), x_{i,n})}{h(i, \mathcal{L}(X_{i,n}'), k)} \) then restart from Line 1
6: return \( x \)
```

**Proposition 1** Algorithm 9 samples points according to the distribution in (20).

The proof is straightforward, and uses the conditional independence of the rows given the column sums are satisfied.
In the most general case when even the columns cannot be simulated using Algorithm 8 or another variant, we apply PDC deterministic second half to both the columns and the rows, which simply demands in the continuous case that there is at least one random variable with a bounded density per column (resp., row). In Algorithm 10 below, each column has a rejection function $t_j$, which is either the normalization of the probability mass function

$$t_j(a) = \frac{\mathbb{P}(X_{ij,j} = a)}{\max_\ell \mathbb{P}(X_{ij,j} = \ell)}$$

or the normalization of the probability density function

$$t_j(a) = \frac{f_{X_{ij,j}}(a)}{\sup_\ell f_{X_{ij,j}}(\ell)}.$$

There is also a row rejection function $s_i$. Let $X'_{i,j}$ denote the random variable with distribution $\mathcal{L}(X_{i,j} | \sum_{\ell=1}^m X_{\ell,j})$. When $\mathcal{L}(X'_{i,j})$ is discrete, we have

$$s_i(a) = \frac{\mathbb{P}(X'_{i,j} = a)}{\max_\ell \mathbb{P}(X'_{i,j} = \ell)}$$

and when $\mathcal{L}(X'_{i,j})$ is continuous, we have

$$s_i(a) = \frac{f_{X'_{i,j}}(a)}{\sup_\ell f_{X'_{i,j}}(\ell)}.$$

Algorithm 10 Generating $\mathcal{L}((X_{1,1}, X_{1,2}, \ldots, X_{m,n}) | E)$

1: let $j' \in \{1, \ldots, n\}$ denote a column.
2: for $j \in \{1, 2, \ldots, n\} \setminus \{j'\}$ do
3:    let $i_j \in \{1, \ldots, m\}$ denote a row in the $j$-th column
4:    for $i = \{1, 2, \ldots, m\} \setminus \{i_j\}$ do
5:        generate $x_{ij}$ from $\mathcal{L}(X_{ij})$
6:    end for
7:    let $x_{ij,j} = c_j - \sum_{i \neq i_j} x_{ij}$
8:    with probability $1 - t_j(x_{ij,j})$ restart from Line 2
9: end for
10: for $i = 1, \ldots, m$ do
11:    let $x_{ij,j'} = r_i - \sum_{j \neq j'} x_{ij}$
12:    with probability $1 - s_i(x_{ij,j'})$ restart from Line 2
13:    if $x_{ij,j'} < 0$ restart from Line 1
14: end for
15: return $x$
**Proposition 2** Algorithm 10 samples points according to the distribution in (20).

The proof is a straightforward application of PDC deterministic second half.

**6 Conclusion**

We presented several novel probabilistic divide-and-conquer algorithms for the random sampling of contingency tables. In the general case, the uniqueness of our approach was in exploiting the independence of bits in a geometric random variable. In the $2 \times n$ case, we exploited the property that two i.i.d. geometric random variables conditioned on their sum equalling $k$ is equal in distribution to $(V, k − V)$, where $V$ is uniformly distributed over the set $\{0, 1, \ldots, k\}$. We also demonstrated other various ways in which a probabilistic divide-and-conquer approach can be fashioned.

It would be of interest to explore the set of parameter values for which the approximate sampling algorithm well-approximates an exact sample, as well as those parameter values for which it performs poorly. The successful resolution of Conjecture 1 would enable exact sampling in the general $m \times n$ setting, as well as the means with which to quantify the bias of the approximate sampling algorithms. In lieu of Conjecture 1, an analysis of the weakly dependent random variables in the probabilistic interpretation would aid in understanding the bias introduced in the approximate sampling algorithms. Finally, while we have fashioned several probabilistic divide-and-conquer algorithms in this paper, it would be of interest to explore other probabilistic divide-and-conquer strategies which may yield more tractable analysis.

**Acknowledgements** The authors gratefully acknowledge helpful discussions with Chris Anderson, Richard Arratia, Jesus de Loera, and also Igor Pak for help with the literature.

**References**

Arratia R, DeSalvo S (2016) Probabilistic divide-and-conquer: a new exact simulation method, with integer partitions as an example. Comb Probab Comput 25(3):324–351

Arratia R, Tavaré S (1994) Independent process approximations for random combinatorial structures. Adv Math 104(1):90–154

Baldoni-Silva W, De Loera JA, Vergne M (2004) Counting integer flows in networks. Found Comput Math 4(3):277–314

Barvinok A (2007) Brunn–Minkowski inequalities for contingency tables and integer flows. Adv Math 211(1):105–122

Barvinok A (2008) Enumerating contingency tables via random permanents. Comb Probab Comput 17(1):1–19

Barvinok A (2009a) Asymptotic estimates for the number of contingency tables, integer flows, and volumes of transportation polytopes. Int Math Res Not IMRN 2:348–385

Barvinok A (2009b) Asymptotic estimates for the number of contingency tables, integer flows, and volumes of transportation polytopes. Int Math Res Not 2009(2):348–385

Barvinok A (2010) What does a random contingency table look like? Comb Probab Comput 19(04):517–539

Barvinok A, Hartigan J (2010) Maximum entropy gaussian approximations for the number of integer points and volumes of polytopes. Adv Appl Math 45(2):252–289

Barvinok A, Hartigan JA (2012) An asymptotic formula for the number of non-negative integer matrices with prescribed row and column sums. Trans Am Math Soc 364(8):4323–4368
Barvinok A, Luria Z, Samorodnitsky A, Yong A (2010) An approximation algorithm for counting contingency tables. Random Struct Algorithms 37(1):25–66
Bender EA (1974) The asymptotic number of non-negative integer matrices with given row and column sums. Discrete Math 10:217–223
Bezáková I, Sinclair A, Štefankovič D, Vigoda E (2006) Negative examples for sequential importance sampling of binary contingency tables. In: Algorithms—ESA 2006. Springer, pp 136–147
Blitzstein J, Diaconis P (2011) A sequential importance sampling algorithm for generating random graphs with prescribed degrees. Internet Math 6(4):489–522
Canfield ER, McKay BD (2005) Asymptotic enumeration of dense 0–1 matrices with equal row sums and equal column sums. J Comb 12(2):R29
Chen Y, Diaconis P, Holmes SP, Liu JS (2005) Sequential Monte Carlo methods for statistical analysis of tables. J Am Stat Assoc 100(469):109–120
Chen Y, Dinwoodie IH, Sullivant S (2006) Sequential importance sampling for multiway tables. Ann Stat 34(1):523–545
Cryan M, Dyer M (2003) A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant. J Comput Syst Sci 67(2):291–310
Cryan M, Dyer M, Goldberg LA, Jerrum M, Martin R (2006) Rapidly mixing Markov chains for sampling contingency tables with a constant number of rows. SIAM J Comput 36(1):247–278
De Loera JA, Hemmecke R, Taurer J, Yoshida R (2004) Effective lattice point counting in rational convex polytopes. J Symb Comput 38(4):1273–1302
DeSalvo S (2018) Probabilistic divide-and-conquer: deterministic second half. Adv Appl Math 92:17–50
Devroye L (1986) Nonuniform random variate generation. Springer, New York
Duchon P (2011) Random generation of combinatorial structures: Boltzmann samplers and beyond. In: Proceedings of the 2011 winter simulation conference (WSC), pp 120–132
Duchon P, Flajolet P, Louchard G, Schaeffer G (2004) Boltzmann samplers for the random generation of combinatorial structures. Comb Probab Comput 13(4–5):577–625
Dyer M, Greenhill C (2000) Polynomial-time counting and sampling of two-rowed contingency tables. Theor Comput Sci 246(1):265–278
Dyer M, Frieze A, Kannan R (1991) A random polynomial-time algorithm for approximating the volume of convex bodies. J ACM 38(1):1–17
Fishman GS (2012) Counting contingency tables via multistage Markov chain Monte Carlo. J Comput Graph Stat 21(3):713–738
Good IJ, Crook JF (1977) The enumeration of arrays and a generalization related to contingency tables. Discrete Math 19(1):23–45
Greenhill C, McKay BD (2008) Asymptotic enumeration of sparse nonnegative integer matrices with specified row and column sums. Adv Appl Math 41(4):459–481
Hall P (1927) The distribution of means for samples of size n drawn from a population in which the variate takes values between 0 and 1, all such values being equally probable. Biometrika 19:240–245
Irwin JO (1927) On the frequency distribution of the means of samples from a population having any law of frequency with finite moments, with special reference to Pearson’s type ii. Biometrika 19:225–239
Kieffer D, Bianchetti L, Poch O (2012) Perfect sampling on 2x2x...x2xkc contingency tables with an application to SAGE data. J Stat Plan Inference 142(4):896–901
Kijima S, Matsui T (2006) Polynomial time perfect sampling algorithm for two-rowed contingency tables. Random Struct Algorithms 29(2):243–256
Knuth DE, Yao AC (1976) The complexity of nonuniform random number generation. In: Algorithms and complexity (Proc. Sympos., Carnegie-Mellon Univ., Pittsburgh, PA, 1976). Academic Press, New York, pp 357–428
Loera JAD, Onn S (2006) All linear and integer programs are slim 3-way transportation programs. SIAM J Optim 17(3):806–821
Matsui T, Matsui Y, Ono Y (2004) Random generation of 2x2x...x2xjk contingency tables. Theor Comput Sci 326(1–3):117–135
Morris BJ (2002) Improved bounds for sampling contingency tables. Random Struct Algorithms 21(2):135–146
Propp JG, Wilson DB (1996) Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Struct Algorithms 9(1–2):223–252
Soules GW (2003) New permanental upper bounds for nonnegative matrices. Linear Multilinear Algebra 51(4):319–337
Steutel F, Thiemann J (1987) On the independence of integer and fractional parts. University of Technology, Department of Mathematics and Computing Science
Von Neumann J (1951) Various techniques used in connection with random digits. J Res Natl Bur Stand Appl Math Ser 13(3):36–38
Wicker N (2010) Perfect sampling algorithm for small m × n contingency tables. Stat Comput 20(1):57–61
Yoshida R, Xi J, Wei S, Zhou F, Haws D (2011) Semigroups and sequential importance sampling for multiway tables. arXiv preprint arXiv:1111.6518

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