Exact Enumeration and Sampling of Matrices with Specified Margins

Jeffrey W. Miller * Matthew T. Harrison †

Abstract

We describe a dynamic programming algorithm for exact counting and exact uniform sampling of matrices with specified row and column sums. The algorithm runs in polynomial time when the column sums are bounded. Binary or non-negative integer matrices are handled. The method is distinguished by applicability to non-regular margins, tractability on large matrices, and the capacity for exact sampling.

Keywords: bipartite graphs, specified degrees, exact counting, exact sampling, tables

1 Introduction

Let $N(\mathbf{p}, \mathbf{q})$ be the number of $m \times n$ binary matrices with margins (row and column sums) $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{N}^m$, $\mathbf{q} = (q_1, \ldots, q_n) \in \mathbb{N}^n$ respectively, and let $M(\mathbf{p}, \mathbf{q})$ be the corresponding number of $\mathbb{N}$-valued matrices. In this paper we develop a technique for efficiently finding $N(\mathbf{p}, \mathbf{q})$ and $M(\mathbf{p}, \mathbf{q})$. Uniform sampling from these sets of matrices is an important problem in statistics [7], and the method given here permits efficient exact uniform sampling once the underlying enumeration problem has been solved.

Since a bipartite graph with degree sequences $\mathbf{p} = (p_1, \ldots, p_m) \in \mathbb{N}^m$, $\mathbf{q} = (q_1, \ldots, q_n) \in \mathbb{N}^n$ (and m, n vertices in each part respectively) can be viewed as a $m \times n$ matrix with row and column sums $(\mathbf{p}, \mathbf{q})$, our technique applies equally well to counting and uniformly sampling such bipartite graphs. Under this correspondence, simple graphs correspond to binary matrices, and multigraphs correspond to $\mathbb{N}$-valued matrices.

*Division of Applied Mathematics, Brown University, Providence, RI 02912. Email: jeffrey_miller at brown.edu. Research supported by a NDSEG fellowship and in part by NSF award DMS-1007593.

†Division of Applied Mathematics, Brown University, Providence, RI 02912. Research supported by NSF award DMS-1007593.

The distinguishing characteristic of the method is its tractability on matrices of non-trivial size. In general, computing $M(\mathbf{p},\mathbf{q})$ is #P-complete [10], and perhaps $N(\mathbf{p},\mathbf{q})$ is as well. However, if we assume a bound on the column sums then our algorithm computes both numbers in polynomial time. After enumeration, uniform samples may be drawn in polynomial expected time for bounded column sums. To our knowledge, all previous algorithms *for the non-regular case* require super-polynomial time (in the worst case) to compute these numbers, even for bounded column sums. (We assume a description length of at least $m+n$ and no more than $m\log a + n\log b$, where $a = \max p_i$, $b = \max q_i$.) In general (without assuming a bound on the column sums), our algorithm computes $N(\mathbf{p},\mathbf{q})$ or $M(\mathbf{p},\mathbf{q})$ in $\mathrm{O}(m(ab+c)(a+b)^{b-1}(b+c)^{b-1}(\log c)^3)$ time for $m \times n$ matrices, where $a = \max p_i$, $b = \max q_i$, and $c = \sum p_i = \sum q_i$. After enumeration, uniform samples may be drawn in $\mathrm{O}(mc\log c)$ expected time.

In complement to most approaches to computing $M(\mathbf{p},\mathbf{q})$, which are efficient for small matrices with large margins, our algorithm is efficient for large matrices with small margins. For instance, in Section 4 we count the 100×100 matrices with margins $(70,30,20,10,5^{(6)},4^{(10)},3^{(20)},2^{(60)})$, $(4^{(80)},3^{(20)})$ (where $x^{(n)}$ denotes x repeated n times).

To illustrate the problem at hand, consider a trivial example: if $\mathbf{p} = (2,2,1,1)$, $\mathbf{q} = (3,2,1)$, then $N(\mathbf{p},\mathbf{q}) = 8$ and $M(\mathbf{p},\mathbf{q}) = 24$. The 8 binary matrices are below.

$$\begin{array}{ccc} 1&1&0\\1&1&0\\1&0&0\\0&0&1 \end{array} \quad \begin{array}{ccc} 1&1&0\\1&1&0\\0&0&1\\1&0&0 \end{array} \quad \begin{array}{ccc} 1&1&0\\1&0&1\\1&0&0\\0&1&0 \end{array} \quad \begin{array}{ccc} 1&1&0\\1&0&1\\0&1&0\\1&0&0 \end{array} \quad \begin{array}{ccc} 1&1&0\\0&1&1\\1&0&0\\1&0&0 \end{array} \quad \begin{array}{ccc} 1&0&1\\1&1&0\\1&0&0\\0&1&0 \end{array} \quad \begin{array}{ccc} 1&0&1\\1&1&0\\0&1&0\\1&0&0 \end{array} \quad \begin{array}{ccc} 0&1&1\\1&1&0\\1&0&0\\1&0&0 \end{array}$$

The paper will proceed as follows:

2 Main results: Recursions, Bounds, Algorithms

Introducing the following notation will be useful. Taking $\mathbb{N} := \{0,1,2,\dots\}$, we consider $\mathbb{N}^n$ to be the subset of $\mathbb{N}^\infty := \{(r_1,r_2,\dots) : r_i \in \mathbb{N} \text{ for } i = 1,2,\dots\}$ such that all but the first n components are zero. Let $L : \mathbb{N}^\infty \to \mathbb{N}^\infty$ denote the left-shift map: $L\mathbf{r} = (r_2,\dots,r_n,0,0,\dots)$. Given $\mathbf{r},\mathbf{s} \in \mathbb{N}^n$, let $\mathbf{r}\backslash\mathbf{s} := \mathbf{r} - \mathbf{s} + L\mathbf{s}$, (which may be read as "$\mathbf{r}$ *reduce* $\mathbf{s}$"), let

$$\binom{\mathbf{r}}{\mathbf{s}} := \binom{r_1}{s_1}\cdots\binom{r_n}{s_n},$$

and let $\bar{\mathbf{r}}$ denote the vector of counts, $\bar{\mathbf{r}} := (\bar{r}_1, \bar{r}_2, \dots)$ where $\bar{r}_i := \#\{j : r_j = i\}$. We write $\mathbf{r} \le \mathbf{s}$ if $r_i \le s_i$ for all i. Given $n \in \mathbb{N}$, let $C_n(k) := \{\mathbf{r} \in \mathbb{N}^n : \sum_i r_i = k\}$ be the n-part compositions (including zero) of k, and given $\mathbf{s} \in \mathbb{N}^n$, let $C^{\mathbf{s}}(k) := \{\mathbf{r} \in C_n(k) : \mathbf{r} \le \mathbf{s}\}$. For $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$, define the numbers

$$N(\mathbf{p}, \mathbf{q}) := \#\{\mathbf{X} \in \{0,1\}^{m \times n} : \sum_j x_{ij} = p_i, \sum_i x_{ij} = q_j, \text{ for } 1 \le i \le m, 1 \le j \le n\},$$

$$M(\mathbf{p}, \mathbf{q}) := \#\{\mathbf{X} \in \mathbb{N}^{m \times n} : \sum_j x_{ij} = p_i, \sum_i x_{ij} = q_j, \text{ for } 1 \le i \le m, 1 \le j \le n\}.$$

Since $N(\mathbf{p}, \mathbf{q})$ and $M(\mathbf{p}, \mathbf{q})$ are fixed under permutations of the row sums $\mathbf{p}$ and column sums $\mathbf{q}$, and since zero margins do not affect the number of matrices and can effectively be ignored, then we may define $\bar{N}(\mathbf{p}, \bar{\mathbf{q}}) := N(\mathbf{p}, \mathbf{q})$ and $\bar{M}(\mathbf{p}, \bar{\mathbf{q}}) := M(\mathbf{p}, \mathbf{q})$ without ambiguity. We can now state our main results.

Theorem 2.1 (Recursions) *The number of matrices with margins $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$ is given by*

(1) $\bar{N}(\mathbf{p}, \mathbf{r}) = \sum_{\mathbf{s} \in C^{\mathbf{r}}(p_1)} \binom{\mathbf{r}}{\mathbf{s}} \bar{N}(L\mathbf{p}, \mathbf{r} \backslash \mathbf{s})$ *for binary matrices, and*

(2) $\bar{M}(\mathbf{p}, \mathbf{r}) = \sum_{\mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(p_1)} \binom{\mathbf{r} + L\mathbf{s}}{\mathbf{s}} \bar{M}(L\mathbf{p}, \mathbf{r} \backslash \mathbf{s})$ *for $\mathbb{N}$-valued matrices,*

where $\mathbf{r} = \bar{\mathbf{q}}$, and in (2)*, we sum over all $\mathbf{s}$ such that $\mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(p_1)$.*

Proofs will be given in Section 5. The Gale-Ryser conditions [11, 29] simplify computation of the sum in (1) by providing a necessary and sufficient condition for there to exist a binary matrix with margins $(\mathbf{p}, \mathbf{q})$: if $q'_i := \#\{j : q_j \ge i\}$ and $p_1 \ge \cdots \ge p_m$, then $N(\mathbf{p}, \mathbf{q}) \ne 0$ if and only if $\sum_{i=1}^{j} p_i \le \sum_{i=1}^{j} q'_i$ for all $j < m$ and $\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q'_i$. This is easily translated into a similar condition in terms of $(\mathbf{p}, \bar{\mathbf{q}})$ and $\bar{N}(\mathbf{p}, \bar{\mathbf{q}})$. The following recursive procedure can be used to compute either $N(\mathbf{p}, \mathbf{q})$ or $M(\mathbf{p}, \mathbf{q})$.

Algorithm 2.2 (Enumeration)
Input: $(\mathbf{p}, \bar{\mathbf{q}})$, where $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$ are row and column sums such that $\sum_i p_i = \sum_i q_i$.
Output: $N(\mathbf{p}, \mathbf{q})$ (or $M(\mathbf{p}, \mathbf{q})$), the number of binary (or $\mathbb{N}$-valued) matrices.
Storage: Lookup table of cached results, initialized with $\bar{N}(\mathbf{0}, \mathbf{0}) = 1$ (or $\bar{M}(\mathbf{0}, \mathbf{0}) = 1$).

(1) *If $\bar{N}(\mathbf{p}, \bar{\mathbf{q}})$ is in the lookup table, return the result.*
(2) *In the binary case, if Gale-Ryser gives $\bar{N}(\mathbf{p}, \bar{\mathbf{q}}) = 0$, cache the result and return 0.*
(3) *Evaluate the sum in Theorem 2.1, recursing to step* (1) *for each term.*
(4) *Cache the result and return it.*

Let $T(\mathbf{p},\mathbf{q})$ be the time (number of machine operations) required by Algorithm 2.2 to compute $N(\mathbf{p},\mathbf{q})$ or $M(\mathbf{p},\mathbf{q})$, after performing an $\mathrm{O}(n^3)$ preprocessing step to compute all needed binomial coefficients. (It turns out that computing $M(\mathbf{p},\mathbf{q})$ always takes longer, but the bounds we prove apply to both $N(\mathbf{p},\mathbf{q})$ and $M(\mathbf{p},\mathbf{q})$.) We give a series of bounds on $T(\mathbf{p},\mathbf{q})$ ranging from tighter but more complicated, to more crude but simpler. The bounds will absorb the $\mathrm{O}(n^3)$ pre-computation except for the trivial case when the maximum column sum is 1.

Theorem 2.3 (Bounds) *Suppose* $(\mathbf{p},\mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$, $a = \max p_i$, $b = \max q_i$, *and* $c = \sum p_i = \sum q_i$. *Then*

(1) $T(\mathbf{p},\mathbf{q}) \le \mathrm{O}((ab+c)(\log c)^3 \sum_{i=1}^{m} \binom{p_i+b-1}{b-1}\binom{p_i+\cdots+p_m+b-1}{b-1})$,
(2) $T(\mathbf{p},\mathbf{q}) \le \mathrm{O}(m(ab+c)(a+b)^{b-1}(b+c)^{b-1}(\log c)^3)$,
(3) $T(\mathbf{p},\mathbf{q}) \le \mathrm{O}(mn^{2b-1}(\log n)^3)$ *for bounded* b,
(4) $T(\mathbf{p},\mathbf{q}) \le \mathrm{O}(mn^b(\log n)^3)$ *for bounded* a, b.

Remark Since we may swap the row sums with the column sums without changing the number of matrices, we could use Algorithm 2.2 on $(\mathbf{q},\bar{\mathbf{p}})$ to compute $N(\mathbf{p},\mathbf{q})$ or $M(\mathbf{p},\mathbf{q})$ using $T(\mathbf{q},\mathbf{p})$ operations, which, for example, is $\mathrm{O}(nm^a(\log m)^3)$ for bounded a, b. $T(\mathbf{p},\mathbf{q})$ also depends on the ordering of the row sums $p_1,\ldots,p_m$ as suggested by Theorem 2.3(1), and we find that putting them in decreasing order $p_1 \ge \cdots \ge p_m$ tends to work well. Algorithm 2.2 is typically made significantly more efficient by using the Gale-Ryser conditions, and this is not accounted for in these bounds. Although we observe empirically that this reduces computation tremendously, we do not have a proof of this.

Algorithm 2.2 traverses a directed acyclic graph in which each node represents a distinct set of input arguments to the algorithm, such as $(\mathbf{p},\bar{\mathbf{q}})$. Node $(\mathbf{u},\bar{\mathbf{v}})$ is the child of node $(\mathbf{p},\bar{\mathbf{q}})$ if the algorithm is called (recursively) with arguments $(\mathbf{u},\bar{\mathbf{v}})$ while executing a call with arguments $(\mathbf{p},\bar{\mathbf{q}})$. If the initial input arguments are $(\mathbf{p},\bar{\mathbf{q}})$, then all nodes are descendents of node $(\mathbf{p},\bar{\mathbf{q}})$. Meanwhile, all nodes are ancestors of node $(\mathbf{0},\mathbf{0})$. Note the correspondence between the children of a node $(\mathbf{u},\bar{\mathbf{v}})$ and the compositions $\mathbf{s} \in C^{\bar{\mathbf{v}}}(u_1)$ in the binary case, and $\mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(u_1)$ in the $\mathbb{N}$-valued case, under which $\mathbf{s}$ corresponds with the child $(L\mathbf{u},\bar{\mathbf{v}}\backslash\mathbf{s})$. We also associate with each node its *count*: the number of matrices with the corresponding margins.

As an additional benefit of caching the counts in a lookup table (as in Algorithm 2.2), once the enumeration is complete we obtain an efficient algorithm for uniform sampling from the set of $(\mathbf{p},\mathbf{q})$ matrices (binary or $\mathbb{N}$-valued). It is straightforward to prove that since the counts are exact, the following algorithm yields a sample from the uniform distribution.

Algorithm 2.4 (Sampling)
Input:
· Row and column sums $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$ *such that* $\sum_i p_i = \sum_i q_i$.
· Lookup table of counts generated by Algorithm 2.2 on input $(\mathbf{p}, \bar{\mathbf{q}})$.
Output: A binary (or $\mathbb{N}$*-valued) matrix with margins* $(\mathbf{p}, \mathbf{q})$*, drawn uniformly at random.*

(1) *Initialize* $(\mathbf{u}, \mathbf{v}) \leftarrow (\mathbf{p}, \mathbf{q})$.
(2) *If* $(\mathbf{u}, \mathbf{v}) = (\mathbf{0}, \mathbf{0})$*, exit.*
(3) *Choose a child* $(L\mathbf{u}, \bar{\mathbf{v}} \backslash \mathbf{s})$ *of* $(\mathbf{u}, \bar{\mathbf{v}})$ *with probability proportional to its count times the number of corresponding rows (that is, the rows* $\mathbf{r} \in C^{\mathbf{v}}(u_1)$ *such that* $\overline{\mathbf{v} - \mathbf{r}} = \bar{\mathbf{v}} \backslash \mathbf{s}$*.)*
(4) *Choose a row uniformly among the corresponding rows.*
(5) $(\mathbf{u}, \mathbf{v}) \leftarrow (L\mathbf{u}, \mathbf{v} - \mathbf{r})$.
(6) *Goto* (2)*.*

In step (3), there are $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ corresponding rows $\mathbf{r}$ in the binary case, and $\binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}$ in the $\mathbb{N}$-valued case. In step (4), in the binary case of course we only choose among $\mathbf{r} \in \{0, 1\}^n$. In Section 6 we prove that Algorithm 2.4 takes $\mathrm{O}(mc \log c)$ expected time per sample, where $c = \sum_i p_i$.

3 Brief review

We briefly cover the previous work on this problem. This review is not exhaustive, focusing instead on those results which are particularly significant or closely related to the present work. Let $H_n(r)$ and $H_n^*(r)$ denote $M(\mathbf{p}, \mathbf{q})$ and $N(\mathbf{p}, \mathbf{q})$, respectively, when $\mathbf{p} = \mathbf{q} = (r, \ldots, r) \in \mathbb{N}^n$. The predominant focus has been on the regular cases $H_n(r)$ and $H_n^*(r)$.

Work on counting these matrices goes back at least as far as MacMahon, who applied his expansive theory to find the polynomial for $H_3(r)$ [21] (Vol II, p.161), and developed the theory of Hammond operators, which we will use below. Redfield's theorem [28], inspired by MacMahon, can be used to derive summations for some special cases, such as $H_n(r), H_n^*(r)$ for $r = 2, 3$, and in similar work Read [26, 27] used Pólya theory to derive these summations for $r = 3$. Two beautiful theoretical results must also be mentioned: Stanley [31] proved that for fixed n, $H_n(r)$ is a polynomial in r, and Gessel [12, 13] showed that for fixed r, both $H_n(r)$ and $H_n^*(r)$ are P-recursive in n, vastly generalizing the linear recursions for $H_n(2)$, $H_n^*(2)$ found by Anand, Dumir, and Gupta [1].

We turn next to algorithmic results more closely related to the present work. McKay [22, 5] has demonstrated a coefficient extraction technique for computing $N(\mathbf{p}, \mathbf{q})$ in the semi-regular case (in which $\mathbf{p} = (a, \ldots, a) \in \mathbb{N}^m$ and $\mathbf{q} = (b, \ldots, b) \in \mathbb{N}^n$). To our

knowledge, McKay's is the most efficient method known previously for $N(\mathbf{p}, \mathbf{q})$. By our analysis it requires at least $\Omega(mn^b)$ time for bounded a, b, while the method presented here is $O(mn^b(\log n)^3)$ in this case. Since this latter bound is quite crude, we expect that our method should have comparable or better performance, and indeed empirically we find that typically it is more efficient. If only b is bounded, McKay's algorithm is still $\Omega(mn^b)$, but the bound on our performance increases to $O(mn^{2b-1}(\log n)^3)$, so it is possible that McKay's algorithm will outperform ours in these cases. Nonetheless, it is important to bear in mind that McKay's algorithm is efficient only in the semi-regular case (while our method permits non-regular margins). If neither a nor b is bounded, McKay's method is exponential in b (as is ours).

Regarding $M(\mathbf{p}, \mathbf{q})$, one of the most efficient algorithms known to date is LattE (Lattice point Enumeration) [19], which uses Barvinok's algorithm [2] to count lattice points contained in convex polyhedra. It runs in polynomial time for any fixed dimension, and as a result it can compute $M(\mathbf{p}, \mathbf{q})$ for astoundingly large margins, provided that m and n are small. However, since the computation time grows very quickly with the dimension, LattE is currently inapplicable when m and n are larger than 6. There are similar algorithms [23, 20, 3] that are efficient for small matrices.

In addition, several other algorithms have been presented for finding $N(\mathbf{p}, \mathbf{q})$ (such as [18, 32, 33, 24]) and $M(\mathbf{p}, \mathbf{q})$ (see review [8]) allowing non-regular margins, however, it appears that all are exponential in the size of the matrix, even for bounded margins. While in this work we are concerned solely with exact results, we note that many useful approximations for $N(\mathbf{p}, \mathbf{q})$ and $M(\mathbf{p}, \mathbf{q})$ (in the general case) have been found, as well as approximate sampling algorithms [17, 7, 14, 4, 16].

4 Applications

4.1 Occurrence matrices from ecology

The need to count and sample occurrence matrices (binary matrices indicating observed pairings of elements of two sets) arises in ecology. A standard dataset of this type is "Darwin's finch data", a 13×17 matrix indicating which of 13 species of finches inhabit which of 17 of the Galápagos Islands. The margins of this matrix are (14, 13, 14, 10, 12, 2, 10, 1, 10, 11, 6, 2, 17), (4, 4, 11, 10, 10, 8, 9, 10, 8, 9, 3, 10, 4, 7, 9, 3, 3). We count the number of such matrices to be 67,149,106,137,567,626 (in 1.5 seconds) confirming [7]. Further, we sample exactly from the uniform distribution over this set at a rate of 0.001 seconds per sample. (All computations were performed on a 64-bit 2.8 GHz machine with 6 GB of RAM.) A similar dataset describes the distribution of 23 land birds on the 15 southern islands in the Gulf of California [7, 6]: this binary matrix has margins (14, 14, 14, 12, 5, 13, 9, 11, 11, 11,

11, 11, 7, 8, 8, 7, 2, 4, 2, 3, 2, 2, 2), (21, 19, 18, 19, 14, 15, 12, 15, 12, 12, 12, 5, 4, 4, 1), for which we count 839,926,782,939,601,640 corresponding binary matrices. Counting takes 1 second, and sampling is 0.002 seconds per sample. One more example of this type: for bird species on the California Islands [25] we find that there are 1,360,641,571,195,211,109,388 binary matrices with margins (1, 4, 3, 2, 1, 1, 1, 5, 1, 3, 1, 4, 4, 5, 1, 2, 1, 5, 4, 5, 3, 7, 1, 3, 2, 4, 1, 3, 2, 4, 6), (2, 14, 24, 8, 2, 5, 20, 15) in 4 seconds; samples take 0.003 seconds each. Larger matrices can be handled as well, provided the margins are small. For example, we count 860585058801817078819959949756...000 (459 digits total, see Appendix) 100×100 matrices with margins $(70, 30, 20, 10, 5^{(6)}, 4^{(10)}, 3^{(20)}, 2^{(60)})$, $(4^{(80)}, 3^{(20)})$ (where $x^{(n)}$ denotes x repeated n times) in 46 minutes. We know of no previous algorithm capable of efficiently and exactly counting and sampling from sets such as this.

4.2 Ehrhart polynomials of the Birkhoff polytope

Stanley [31] proved a remarkable conjecture of Anand, Dumir, and Gupta [1]: given $n \in \mathbb{N}$, $H_n(r)$ is a polynomial in r (where $H_n(r) = M(\mathbf{p}, \mathbf{q})$ with $\mathbf{p} = \mathbf{q} = (r, r, \ldots, r) \in \mathbb{N}^n$). Given $H_n(1), \ldots, H_n(\binom{n-1}{2})$, one can solve for the coefficients of $H_n(r)$ (as we describe below). These polynomials have been computed for $n \leq 9$ by Beck and Pixton [3]. As an application of our method, we computed them for $n \leq 8$, and found that the computation time is comparable to that of Beck and Pixton. For $n = 4, \ldots, 8$, the numbers $H_n(r)$ for $r = 1, \ldots, \binom{n-1}{2}$ are listed in the Appendix, and the polynomial $H_4(r)$ is displayed here as an example. Our results confirm those of Beck and Pixton.

$$H_4(r) = 1 + (65/18)r + (379/63)r^2 + (35117/5670)r^3 + (43/10)r^4$$
$$+(1109/540)r^5 + (2/3)r^6 + (19/135)r^7 + (11/630)r^8 + (11/11340)r^9.$$

The coefficients of $H_n(r)$ can be determined by the following method. By Stanley's theorem [31], $H_n(r)$ is a polynomial in r such that (a) $\deg H_n(r) = (n-1)^2$, (b) $H_n(-1) = \cdots = H_n(-n+1) = 0$, and (c) $H_n(-n-r) = (-1)^{(n-1)^2} H_n(r)$ for $r \in \mathbb{N}$. For each $n = 4, \ldots, 8$, we perform the following computation. Let $k = \binom{n-1}{2}$ and $d = (n-1)^2$. Compute the numbers $H_n(r)$ for $r = 0, 1, \ldots, k$ using Algorithm 2.2, and form the vector $\mathbf{v} := (H_n(-n-k+1), \ldots, H_n(k))^\top \in \mathbb{Z}^{d+1}$ using (b) and (c). Form the matrix $A = \big((i-n-k)^{j-1}\big)_{i,j=1}^{d+1} \in \mathbb{Z}^{(d+1)\times(d+1)}$, and compute $\mathbf{u} = A^{-1}\mathbf{v}$. Then by (a),

$$H_n(r) = \sum_{j=0}^{d} u_{j+1} r^j.$$

4.3 Contingency Tables

As an example of counting contingency tables with non-regular margins, we count 6200174883910495922978969565 31...000 (483 digits total, see Appendix) 100×100 matrices with margins $(70, 30, 20, 10, 5^{(6)}, 4^{(10)}, 3^{(20)}, 2^{(60)})$, $(4^{(80)}, 3^{(20)})$ (where $x^{(n)}$ denotes x repeated n times) in 118 minutes. Again, we know of no previous algorithm capable of efficiently and exactly counting and sampling from sets such as this. (However, for small contingency tables with large margins, our algorithm is much less efficient than other methods such as LattE.) Exact uniform sampling is possible for contingency tables as well, which occasionally finds use in statistics [9].

5 Proof of recursions

We give two proofs of Theorem 2.1. The first is a "direct" proof, which provides the basis for the sampling algorithm outlined above. In addition to the direct proof, we also provide a proof using generating functions which is seen to be a natural consequence of MacMahon's development [21] of symmetric functions, and yields results of a more general nature.

5.1 Preliminary observations

For $\mathbf{r} \in \mathbb{N}^n$, let $\mathbf{r}'$ denote the conjugate of $\mathbf{r}$, that is, $r'_i = \#\{j : r_j \geq i\}$ for $i = 1, 2, 3, \dots$. For $\mathbf{r}, \mathbf{s} \in \mathbb{N}^\infty$, let $\mathbf{r} \wedge \mathbf{s}$ denote the component-wise minimum, that is, $(r_1 \wedge s_1, r_2 \wedge s_2, \dots)$. In particular, $\mathbf{r} \wedge \mathbf{1} = (r_1 \wedge 1, r_2 \wedge 1, \dots)$. Recall our convention that $\mathbb{N}^n$ is considered to be the subset of $\mathbb{N}^\infty$ such that all but the first n components are zero. (Similarly, we consider $\mathbb{Z}^n \subset \mathbb{Z}^\infty$.)

Lemma 5.1 *Let* $\mathbf{u}, \mathbf{v} \in \mathbb{N}^n$ *such that* $\mathbf{u} \leq \mathbf{v}$.

(1) *Suppose* $\mathbf{s} \in \mathbb{N}^d$ *for some* $d \in \mathbb{N}$. *Then* $\overline{\mathbf{v} - \mathbf{u}} = \bar{\mathbf{v}} \backslash \mathbf{s}$ *if and only if* $\mathbf{s} = \mathbf{v}' - (\mathbf{v} - \mathbf{u})'$.
(2) *If* $\mathbf{s} = \mathbf{v}' - (\mathbf{v} - \mathbf{u})'$ *then* $\sum s_i = \sum u_i$ *and* $\mathbf{s} \leq \bar{\mathbf{v}} + L\mathbf{s}$.
(3) *If* $\mathbf{s} = \mathbf{v}' - (\mathbf{v} - \mathbf{u})'$ *and* $\mathbf{u} \leq \mathbf{v} \wedge \mathbf{1}$ *then* $\mathbf{s} \leq \bar{\mathbf{v}}$.

Proof (1) Letting I be the identity operator, a straightforward calculation shows that for any $d \in \mathbb{N}$, $\mathbf{r} \in \mathbb{Z}^d$, we have $\left(\sum_{k=0}^\infty L^k\right)(I - L)\mathbf{r} = \mathbf{r}$ and $(I - L)\left(\sum_{k=0}^\infty L^k\right)\mathbf{r} = \mathbf{r}$, that is, $(I - L)^{-1} = \sum_{k=0}^\infty L^k$ on $\mathbb{Z}^d$, where I is the identity operator. Further, $(I - L)^{-1}\bar{\mathbf{r}} = \mathbf{r}'$. Thus, $\overline{\mathbf{v} - \mathbf{u}} = \bar{\mathbf{v}} \backslash \mathbf{s} = \bar{\mathbf{v}} - \mathbf{s} + L\mathbf{s}$ if and only if $(I - L)\mathbf{s} = \bar{\mathbf{v}} - \overline{\mathbf{v} - \mathbf{u}}$ if and only if $\mathbf{s} = (I - L)^{-1}(\bar{\mathbf{v}} - \overline{\mathbf{v} - \mathbf{u}}) = \mathbf{v}' - (\mathbf{v} - \mathbf{u})'$.

(2) If $\mathbf{s} = \mathbf{v}' - (\mathbf{v}-\mathbf{u})'$ then $\sum s_i = \sum v'_i - \sum(\mathbf{v}-\mathbf{u})'_i = \sum v_i - \sum(v_i - u_i) = \sum u_i$ since $\sum r'_i = \sum r_i$ for all $\mathbf{r} \in \mathbb{N}^n$. By (1), $\bar{\mathbf{v}} - \mathbf{s} + L\mathbf{s} = \bar{\mathbf{v}} \backslash \mathbf{s} = \overline{\mathbf{v}-\mathbf{u}} \geq \mathbf{0}$ and so $\mathbf{s} \leq \bar{\mathbf{v}} + L\mathbf{s}$.

(3) If $\mathbf{s} = \mathbf{v}' - (\mathbf{v}-\mathbf{u})'$ and $\mathbf{u} \leq \mathbf{v} \wedge \mathbf{1}$ then by definition, $s_i = \#\{j : v_j \geq i\} - \#\{j : v_j - u_j \geq i\} = \#\{j : v_j = i \text{ and } u_j = 1\} \leq \#\{j : v_j = i\} = \bar{v}_i$. ∎

Lemma 5.2 *Let* $\mathbf{v} \in \mathbb{N}^n$, $k \in \mathbb{N}$, *and let* $f(\mathbf{u}) = \mathbf{v}' - (\mathbf{v}-\mathbf{u})'$ *for* $\mathbf{u} \in \mathbb{N}^n$ *such that* $\mathbf{u} \leq \mathbf{v}$.

(1) $f(C^{\mathbf{v}\wedge\mathbf{1}}(k)) = C^{\bar{\mathbf{v}}}(k)$, *and for any* $\mathbf{s} \in C^{\bar{\mathbf{v}}}(k)$, $\#\{\mathbf{u} \in C^{\mathbf{v}\wedge\mathbf{1}}(k) : f(\mathbf{u}) = \mathbf{s}\} = \binom{\bar{\mathbf{v}}}{\mathbf{s}}$.
(2) $f(C^{\mathbf{v}}(k)) = \{\mathbf{s} : \mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(k)\}$, *and for any* $\mathbf{s}$ *such that* $\mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(k)$, $\#\{\mathbf{u} \in C^{\mathbf{v}}(k) : f(\mathbf{u}) = \mathbf{s}\} = \binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}$.

Proof (1) $f(C^{\mathbf{v}\wedge\mathbf{1}}(k)) \subset C^{\bar{\mathbf{v}}}(k)$ follows from Lemma 5.1(2 and 3). Let $\mathbf{s} \in C^{\bar{\mathbf{v}}}(k)$. Choose $\mathbf{u}$ as follows. For $i = 1, 2, 3, \ldots$, choose s_i of the $\bar{v}_i$ positions j such that $v_j = i$, and set $u_j = 1$ for each chosen j. (Set $u_j = 0$ for all remaining j.) This determines some $\mathbf{u} \in C^{\mathbf{v}\wedge\mathbf{1}}(k)$ such that $s_i = \#\{j : v_j = i \text{ and } u_j = 1\}$ for all i. Furthermore, it is not hard to see that any such $\mathbf{u}$ is obtained by such a sequence of choices. Now, as in the proof of Lemma 5.1(3), $s_i = \#\{j : v_j = i \text{ and } u_j = 1\}$ if and only if $f(\mathbf{u}) = \mathbf{s}$ (when $\mathbf{u} \leq \mathbf{v} \wedge \mathbf{1}$). Hence, $f(C^{\mathbf{v}\wedge\mathbf{1}}(k)) \supset C^{\bar{\mathbf{v}}}(k)$, and since there were $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ possible ways to choose $\mathbf{u}$, then this proves (1).

(2) $f(C^{\mathbf{v}}(k)) \subset \{\mathbf{s} : \mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(k)\}$ follows from Lemma 5.1(2). Suppose $\mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(k)$. Let $\mathbf{r} = \bar{\mathbf{v}}$ and $\mathbf{t} = \mathbf{r}\backslash\mathbf{s}$. Note that $\mathbf{t} \geq \mathbf{0}$ since $\mathbf{s} \leq \mathbf{r} + L\mathbf{s}$. Also, $\mathbf{r}, \mathbf{s}, \mathbf{t} \in \mathbb{N}^d$ where $d = \max v_j$. Choose $\mathbf{u}$ as follows. First, consider the r_d positions j in $\mathbf{v}$ at which $v_j = d$. There are $\binom{r_d}{t_d}$ ways to choose t_d of these r_d positions. Having made such a choice, we set $u_j = v_j - d = 0$ for each such j that was chosen. Next, consider the r_{d-1} positions j at which $v_j = d-1$, in addition to the $r_d - t_d$ remaining positions at which $v_j = d$. There are $\binom{r_{d-1}+(r_d-t_d)}{t_{d-1}}$ ways to choose t_{d-1} of these. Having made such a choice, we set $u_j = v_j - (d-1)$ for each such j that was chosen. Continuing in this way, for $i = d-2, \ldots, 1$: consider the r_i positions j in $\mathbf{v}$ which $v_j = i$, in addition to the $r_{i+1} + \cdots + r_d - t_d - \cdots - t_{i+1}$ remaining positions at which $v_j > i$, choose t_i of these (in one of $\binom{r_i + r_{i+1} + \cdots + r_d - t_d - \cdots - t_{i+1}}{t_i}$ ways), and set $u_j = v_j - i$ for each such j that was chosen. After following these steps for each i, set $u_j = v_j$ for any remaining positions j. This determines some $\mathbf{u}$ such that $\mathbf{0} \leq \mathbf{u} \leq \mathbf{v}$.

Now, for $i = d, d-1, \ldots, 1$, we have chosen t_i positions j and we have set $u_j = v_j - i$. That is, $t_i = \#\{j : v_j - u_j = i\}$, and so $\mathbf{t} = \overline{\mathbf{v}-\mathbf{u}}$. Hence, $\overline{\mathbf{v}-\mathbf{u}} = \bar{\mathbf{v}}\backslash\mathbf{s}$ (by the definition of $\mathbf{t}$), so $\mathbf{s} = f(\mathbf{u})$ by Lemma 5.1(1), and additionally, $\sum u_j = \sum s_j = k$ by 5.1(2). Thus, we have shown that $f(C^{\mathbf{v}}(k)) \supset \{\mathbf{s} : \mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(k)\}$.

Using $t_j = r_j - s_j + s_{j+1}$ (the definition of $\mathbf{t}$), we see that there were

$$\binom{r_d}{t_d}\binom{r_{d-1}+(r_d-t_d)}{t_{d-1}}\cdots\binom{r_1+r_2+\cdots+r_d-t_d-\cdots-t_2}{t_1}$$

$$=\binom{r_d}{s_d}\binom{r_{d-1}+s_d}{s_{d-1}}\cdots\binom{r_1+s_2}{s_1}=\binom{\mathbf{r}+L\mathbf{s}}{\mathbf{s}}>0$$

ways to make such a sequence of choices, where the inequality holds since $\mathbf{s} \leq \mathbf{r} + L\mathbf{s}$. Hence, there are at least $\binom{\mathbf{r}+L\mathbf{s}}{\mathbf{s}}$ distinct choices of $\mathbf{u} \in C^{\mathbf{v}}(k)$ such that $f(\mathbf{u}) = \mathbf{s}$. On the other hand, given any $\mathbf{u} \in C^{\mathbf{v}}(k)$ such that $f(\mathbf{u}) = \mathbf{s}$, we have $\mathbf{t} = \overline{\mathbf{v} - \mathbf{u}}$ (by Lemma 5.1(1)), thus $t_i = \#\{j : u_j = v_j - i\}$, and since $v_j \geq i$ for any j such that $u_j = v_j - i$, such a $\mathbf{u}$ is obtained by one of the sequences of choices above. Hence, $\#\{\mathbf{u} \in C^{\mathbf{v}}(k) : f(\mathbf{u}) = \mathbf{s}\} = \binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}$. ∎

5.2 Direct proof

We are now prepared to prove Theorem 2.1. Recall the statement of the theorem:

The number of matrices with margins $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$ *is given by*

(1) $\bar{N}(\mathbf{p}, \mathbf{r}) = \sum_{\mathbf{s}\in C^{\mathbf{r}}(p_1)} \binom{\mathbf{r}}{\mathbf{s}} \bar{N}(L\mathbf{p}, \mathbf{r}\backslash\mathbf{s})$ *for binary matrices, and*

(2) $\bar{M}(\mathbf{p}, \mathbf{r}) = \sum_{\mathbf{s}\in C^{\mathbf{r}+L\mathbf{s}}(p_1)} \binom{\mathbf{r}+L\mathbf{s}}{\mathbf{s}} \bar{M}(L\mathbf{p}, \mathbf{r}\backslash\mathbf{s})$ *for* $\mathbb{N}$*-valued matrices,*

where $\mathbf{r} = \bar{\mathbf{q}}$*, and in* (2)*, we sum over all* $\mathbf{s}$ *such that* $\mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(p_1)$.

Proof of Theorem 2.1

(1) First, we prove the binary case. Let $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$, $\mathbf{r} = \bar{\mathbf{q}}$. Using Lemma 5.2(1), define the surjection $f : C^{\mathbf{q}\wedge\mathbf{1}}(p_1) \to C^{\mathbf{r}}(p_1)$ by $f(\mathbf{u}) = \mathbf{q}' - (\mathbf{q} - \mathbf{u})'$. Then

$$\bar{N}(\mathbf{p}, \mathbf{r}) = N(\mathbf{p}, \mathbf{q}) \overset{\text{(a)}}{=} \sum_{\mathbf{u}\in C^{\mathbf{q}\wedge\mathbf{1}}(p_1)} N(L\mathbf{p}, \mathbf{q} - \mathbf{u})$$

$$\overset{\text{(b)}}{=} \sum_{\mathbf{s}\in C^{\mathbf{r}}(p_1)} \sum_{\mathbf{u}\in f^{-1}(\mathbf{s})} N(L\mathbf{p}, \mathbf{q} - \mathbf{u}) \overset{\text{(c)}}{=} \sum_{\mathbf{s}\in C^{\mathbf{r}}(p_1)} \binom{\mathbf{r}}{\mathbf{s}} \bar{N}(L\mathbf{p}, \mathbf{r}\backslash\mathbf{s}).$$

Step (a) follows from partitioning the set of $(\mathbf{p}, \mathbf{q})$ matrices according to the first row $\mathbf{u} \in C^{\mathbf{q}\wedge\mathbf{1}}(p_1)$ of the matrix. Step (b) partitions $C^{\mathbf{q}\wedge\mathbf{1}}(p_1)$ into the level sets of f, that is, the sets $f^{-1}(\mathbf{s}) = \{\mathbf{u} \in C^{\mathbf{q}\wedge\mathbf{1}}(p_1) : f(\mathbf{u}) = \mathbf{s}\}$ as $\mathbf{s}$ ranges over $f(C^{\mathbf{q}\wedge\mathbf{1}}(p_1)) = C^{\mathbf{r}}(p_1)$. Step (c) follows since if $f(\mathbf{u}) = \mathbf{s}$ then $\overline{\mathbf{q} - \mathbf{u}} = \mathbf{r}\backslash\mathbf{s}$ (by Lemma 5.1(1)) and thus $N(L\mathbf{p}, \mathbf{q} - \mathbf{u}) = \bar{N}(L\mathbf{p}, \mathbf{r}\backslash\mathbf{s})$, and since $\#f^{-1}(\mathbf{s}) = \binom{\mathbf{r}}{\mathbf{s}}$ (by Lemma 5.2(1)) . This proves 2.1(1).

(2) Now, we consider the $\mathbb{N}$-valued case. Let $S = \{\mathbf{s} : \mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(p_1)\}$. Using Lemma 5.2(2), define the surjection $g : C^{\mathbf{q}}(p_1) \to S$ by $g(\mathbf{u}) = \mathbf{q}' - (\mathbf{q} - \mathbf{u})'$. Then, similarly,

$$\bar{M}(\mathbf{p}, \mathbf{r}) = M(\mathbf{p}, \mathbf{q}) \stackrel{\text{(a)}}{=} \sum_{\mathbf{u} \in C^{\mathbf{q}}(p_1)} M(L\mathbf{p}, \mathbf{q} - \mathbf{u})$$

$$\stackrel{\text{(b)}}{=} \sum_{\mathbf{s} \in S} \sum_{\mathbf{u} \in g^{-1}(\mathbf{s})} M(L\mathbf{p}, \mathbf{q} - \mathbf{u}) \stackrel{\text{(c)}}{=} \sum_{\mathbf{s} \in S} \binom{\mathbf{r} + L\mathbf{s}}{\mathbf{s}} \bar{M}(L\mathbf{p}, \mathbf{r} \backslash \mathbf{s}).$$

As before, step (a) follows from partitioning the set of matrices according to the first row $\mathbf{u} \in C^{\mathbf{q}}(p_1)$, step (b) partitions $C^{\mathbf{q}}(p_1)$ into the level sets of g, and step (c) follows since $\#g^{-1}(\mathbf{s}) = \binom{\mathbf{r}+L\mathbf{s}}{\mathbf{s}}$ (by Lemma 5.2(2)). This proves Theorem 2.1. ∎

5.3 Generating function proof

In addition to the direct approach above, one may also view the recursions as the application of a certain differential operator to a certain symmetric functions. Although such operators were used extensively by MacMahon [21] on problems of this type, at first it would appear that for computation this approach would be hopelessly inefficient in all but the simplest examples. In fact, it turns out that a simple observation allows one to exploit regularities in the present problem, reducing the computation time to polynomial for bounded margins. Specifically, when there are many columns with the same sum, the symmetric function under consideration has many repeated factors, and the action of the operator in this situation takes a simplified form.

We will identify $N(\mathbf{p}, \mathbf{q})$ and $M(\mathbf{p}, \mathbf{q})$ as the coefficients of certain symmetric functions, introduce an operator for extracting coefficients, and show that its action yields the recursion above.

Let e_n denote the elementary symmetric function of degree n, in a countably infinite number of variables $\{x_1, x_2, \dots\}$:

$$e_n := \sum_{r_1 < r_2 < \cdots < r_n} x_{r_1} x_{r_2} \cdots x_{r_n},$$

and let h_n be the complete symmetric function of degree n:

$$h_n := \sum_{r_1 \le r_2 \le \cdots \le r_n} x_{r_1} x_{r_2} \cdots x_{r_n},$$

where $r_1, \dots, r_n \in \{1, 2, 3, \dots\}$. For convenience, let $x_0 = e_0 = h_0 = 1$ and $e_n = h_n = 0$ if $n < 0$. Given $\mathbf{r} \in \mathbb{N}^n$, let $x^{\mathbf{r}} := x_1^{r_1} \cdots x_n^{r_n}$ and $x_{\mathbf{r}} := x_{r_1} \cdots x_{r_n}$. Apply the same notation for $e^{\mathbf{r}}$ and $e_{\mathbf{r}}$, as well as $h^{\mathbf{r}}$ and $h_{\mathbf{r}}$. Note that if $\mathbf{r} = \bar{\mathbf{q}}$, then $x^{\mathbf{r}} = x_{\mathbf{q}}$.

Lemma 5.3 (MacMahon) *For any* $\mathbf{p} \in \mathbb{N}^m$, $\mathbf{q} \in \mathbb{N}^n$,

(1) $N(\mathbf{p}, \mathbf{q})$ *is the coefficient of* $x^{\mathbf{p}}$ *in* $e_{\mathbf{q}}$, *and*
(2) $M(\mathbf{p}, \mathbf{q})$ *is the coefficient of* $x^{\mathbf{p}}$ *in* $h_{\mathbf{q}}$.

Proof The coefficient of $x^{\mathbf{p}}$ in $e_{\mathbf{q}}$ is the number of ways to choose one term from each of the n factors $e_{q_1}, \ldots, e_{q_n}$, such that the product of these terms is $x^{\mathbf{p}}$. Observe the correspondence in which the n factors in $e_{\mathbf{q}} = e_{q_1} \cdots e_{q_n}$ are identified with the n columns in the matrix, and choosing a term $x_{\mathbf{r}}$ in a given e_{q_i} corresponds to choosing column i to have ones in rows $r_1, \ldots, r_{q_i}$ (and zeros elsewhere). For any choice of terms $x_{\mathbf{r}^1}, \ldots, x_{\mathbf{r}^n}$ from $e_{q_1}, \ldots, e_{q_n}$ respectively such that $x_{\mathbf{r}^1} \cdots x_{\mathbf{r}^n} = x^{\mathbf{p}}$, we have a binary matrix with margins $(\mathbf{p}, \mathbf{q})$, and conversely, for any such matrix there is such a choice of terms $x_{\mathbf{r}^1}, \ldots, x_{\mathbf{r}^n}$. Thus, the coefficient of $x^{\mathbf{p}}$ in $e_{\mathbf{q}}$ is also the number of such matrices, $N(\mathbf{p}, \mathbf{q})$.

The proof for $M(\mathbf{p}, \mathbf{q})$ is the same, except in this case, choosing a term $x^{\mathbf{r}}$ in a given h_{q_i} corresponds to choosing column i to have entries $r_1, \ldots, r_m$, and a sequence such that $x^{\mathbf{r}^1} \cdots x^{\mathbf{r}^n} = x^{\mathbf{p}}$ corresponds to an $\mathbb{N}$-valued matrix with margins $(\mathbf{p}, \mathbf{q})$. ∎

In what follows, when we say "series", we mean a formal power series in $x_1, x_2, \ldots$. Write $\mathbf{x} = (x_1, x_2, \ldots)$ for the sequence of variables, and let $R\mathbf{x} = (0, x_1, x_2, \ldots)$. For $k \in \mathbb{N}$, define the differential operator:

$$D_k := \frac{1}{k!} \frac{\partial^k}{\partial x_1^k}\Big|_{\mathbf{x}=R\mathbf{x}}.$$

In other words, after taking the kth derivative with respect to x_1 and dividing by $k!$, replace x_1 with zero, and x_{i+1} with x_i for $i = 1, 2, \ldots$. Acting on a series in $x_1, x_2, \ldots$, the operator D_k annihilates every term except those in which the power of x_1 is exactly k. (Note that D_k coincides with Hammond's operator [15, 21], on any symmetric series.) Define

$$D_{\mathbf{r}} := D_{r_n} \cdots D_{r_1}$$

(note the reverse order) where $n = \max\{j : r_j \neq 0\}$ if $\mathbf{r} \neq \mathbf{0}$ and $D_{\mathbf{r}}$ is the identity operator otherwise. By applying the operator $D_{\mathbf{r}}$, we keep only terms exactly divisible by $x^{\mathbf{r}}$ (that is, the power of x_i is r_i for $i = 1, \ldots, n$). In particular, if f is a homogeneous series of degree $\sum r_i$, (so that each term has degree $\sum r_i$), then $D_{\mathbf{r}} f$ is a number equal to the coefficient of $x^{\mathbf{r}}$ in f. Since $e_{\mathbf{q}}$ and $h_{\mathbf{q}}$ are homogeneous series of degree $\sum q_i$, then by Lemma 5.3 we have

Corollary 5.4 *For any* $\mathbf{p} \in \mathbb{N}^m$, $\mathbf{q} \in \mathbb{N}^n$ *such that* $\sum p_i = \sum q_i$,

(1) $D_{\mathbf{p}} e_{\mathbf{q}} = N(\mathbf{p}, \mathbf{q})$,
(2) $D_{\mathbf{p}} h_{\mathbf{q}} = M(\mathbf{p}, \mathbf{q})$. ∎

The following identities begin to reveal the utility of the operators D_k.

Lemma 5.5 (MacMahon) *For $n, k \in \mathbb{N}$,*

(1) $D_k h_n = h_{n-k}$

(2) $D_k e_n = \begin{cases} e_{n-k} & \text{if } k \leq 1 \\ 0 & \text{if } k > 1 \end{cases}$

(3) *For any functions $f_1, \ldots, f_n$,*

$$D_k(f_1 \cdots f_n) = \sum_{\mathbf{s} \in C_n(k)} (D_{s_1} f_1) \cdots (D_{s_n} f_n).$$

Proof (1) and (2) are straightforward calculations. For (3), writing $\partial^k = \frac{\partial^k}{\partial x_1^k}$, we have

$$k! D_k(f_1 \cdots f_n) = \sum_{\mathbf{r} \in \{1,\ldots,n\}^k} (\partial^{\bar{r}_1} f_1) \cdots (\partial^{\bar{r}_n} f_n)\Big|_{\mathbf{x}=R\mathbf{x}}$$

$$= \sum_{\mathbf{s} \in C_n(k)} \frac{k!}{s_1! \ldots s_n!} (\partial^{s_1} f_1) \cdots (\partial^{s_n} f_n)\Big|_{\mathbf{x}=R\mathbf{x}} = \sum_{\mathbf{s} \in C_n(k)} k! (D_{s_1} f_1) \cdots (D_{s_n} f_n),$$

where the first step follows by recursive application of the product rule, and the second by collecting like terms. ∎

Lemma 5.6 (Power rules) *For any $k \in \mathbb{N}$, $\mathbf{r} \in \mathbb{N}^n$,*

(1) $D_k e^{\mathbf{r}} = \sum_{\mathbf{s} \in C^{\mathbf{r}}(k)} \binom{\mathbf{r}}{\mathbf{s}} e^{\mathbf{r} \setminus \mathbf{s}}$

(2) $D_k h^{\mathbf{r}} = \sum_{\mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(k)} \binom{\mathbf{r} + L\mathbf{s}}{\mathbf{s}} h^{\mathbf{r} \setminus \mathbf{s}}.$

Proof (1) For any $m, i \in \mathbb{N}$,

$$D_k e_i^m = \binom{m}{k} e_i^{m-k} e_{i-1}^k$$

by Lemma 5.5(2 and 3). Thus,

$$D_k e^{\mathbf{r}} = D_k(e_1^{r_1} \cdots e_n^{r_n}) \overset{\text{(a)}}{=} \sum_{\mathbf{s} \in C_n(k)} (D_{s_1} e_1^{r_1}) \cdots (D_{s_n} e_n^{r_n})$$

$$\overset{\text{(b)}}{=} \sum_{\mathbf{s}\in C_n(k)} \left(\binom{r_1}{s_1} e_1^{r_1-s_1} e_0^{s_1}\right) \cdots \left(\binom{r_n}{s_n} e_n^{r_n-s_n} e_{n-1}^{s_n}\right)$$

$$\overset{\text{(c)}}{=} \sum_{\mathbf{s}\in C_n(k)} \binom{\mathbf{r}}{\mathbf{s}} e_1^{r_1-s_1+s_2} e_2^{r_2-s_2+s_3} \cdots e_n^{r_n-s_n} \overset{\text{(d)}}{=} \sum_{\mathbf{s}\in C^{\mathbf{r}}(k)} \binom{\mathbf{r}}{\mathbf{s}} e^{\mathbf{r}\backslash\mathbf{s}},$$

with (a) by Lemma 5.5(3), (b) by the preceding observation, (c) by collecting factors, and (d) since $\binom{\mathbf{r}}{\mathbf{s}} = 0$ unless $\mathbf{s} \le \mathbf{r}$ and by the definition of $\mathbf{r}\backslash\mathbf{s}$.

(2) Let $m = \sum r_i$ and let $\mathbf{v} \in \mathbb{N}^m$ be any vector such that $\bar{\mathbf{v}} = \mathbf{r}$, so that $h^{\mathbf{r}} = h_{\mathbf{v}}$. Let $S = \{\mathbf{s} : \mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(k)\}$, and using Lemma 5.2(2), define the surjection $g : C^{\mathbf{v}}(k) \to S$ by $g(\mathbf{u}) = \mathbf{v}' - (\mathbf{v} - \mathbf{u})'$. Then

$$D_k h^{\mathbf{r}} = D_k(h_{v_1} \dots h_{v_m}) \overset{\text{(a)}}{=} \sum_{\mathbf{u}\in C_m(k)} (D_{u_1} h_{v_1}) \dots (D_{u_m} h_{v_m})$$

$$\overset{\text{(b)}}{=} \sum_{\mathbf{u}\in C_m(k)} h_{\mathbf{v}-\mathbf{u}} \overset{\text{(c)}}{=} \sum_{\mathbf{u}\in C^{\mathbf{v}}(k)} h_{\mathbf{v}-\mathbf{u}} = \sum_{\mathbf{u}\in C^{\mathbf{v}}(k)} h^{\overline{\mathbf{v}-\mathbf{u}}}$$

$$\overset{\text{(d)}}{=} \sum_{\mathbf{s}\in C^{\mathbf{r}+L\mathbf{s}}(k)} \sum_{\mathbf{u}\in g^{-1}(\mathbf{s})} h^{\overline{\mathbf{v}-\mathbf{u}}} \overset{\text{(e)}}{=} \sum_{\mathbf{s}\in C^{\mathbf{r}+L\mathbf{s}}(k)} \binom{\mathbf{r}+L\mathbf{s}}{\mathbf{s}} h^{\mathbf{r}\backslash\mathbf{s}},$$

where (a) follows from Lemma 5.5(3), (b) by 5.5(1), (c) since $h_j = 0$ if $j < 0$ and thus $h_{\mathbf{v}-\mathbf{u}} = 0$ if $\mathbf{u} \not\le \mathbf{v}$, (d) by 5.2(2), and (e) by 5.1(1) and 5.2(2). ∎

We now complete the generating function proof of Theorem 2.1. If $\mathbf{p} \in \mathbb{N}^m$, $\mathbf{q} \in \mathbb{N}^n$, $\sum p_i = \sum q_i$, and $\mathbf{r} = \bar{\mathbf{q}}$, then by Lemma 5.6(1),

$$D_{\mathbf{p}} e^{\mathbf{r}} = D_{L\mathbf{p}}(D_{p_1} e^{\mathbf{r}}) = \sum_{\mathbf{s}\in C^{\mathbf{r}}(p_1)} \binom{\mathbf{r}}{\mathbf{s}} D_{L\mathbf{p}} e^{\mathbf{r}\backslash\mathbf{s}},$$

and since $e^{\mathbf{r}} = e_{\mathbf{q}}$, then using Corollary 5.4 (twice) we have

$$\bar{N}(\mathbf{p}, \mathbf{r}) = N(\mathbf{p}, \mathbf{q}) = D_{\mathbf{p}} e_{\mathbf{q}} = D_{\mathbf{p}} e^{\mathbf{r}} = \sum_{\mathbf{s}\in C^{\mathbf{r}}(p_1)} \binom{\mathbf{r}}{\mathbf{s}} \bar{N}(L\mathbf{p}, \mathbf{r}\backslash\mathbf{s}).$$

This proves Theorem 2.1(1). Similarly, in view of Corollary 5.4, Theorem 2.1(2) follows immediately from Lemma 5.6(2).

6 Computation time

Let $W(\mathbf{r}) := \sum_{k=1}^{n} kr_k =$ the *weight* of $\mathbf{r} \in \mathbb{Z}^n$.

Lemma 6.1 (Properties of the weight) *If* $\mathbf{r}, \mathbf{s} \in \mathbb{Z}^n$ *then*

(1) $W(\mathbf{r}+\mathbf{s}) = W(\mathbf{r}) + W(\mathbf{s})$
(2) $W(\mathbf{s} - L\mathbf{s}) = \sum s_i$
(3) $W(\mathbf{r}\backslash\mathbf{s}) = W(\mathbf{r}) - \sum s_i$
(4) $W(\bar{\mathbf{s}}) = \sum s_i$.

Proof All four are simple calculations. ∎

For the rest of this section, fix $(\mathbf{p}, \mathbf{q}) \in \mathbb{N}^m \times \mathbb{N}^n$ such that $\sum p_i = \sum q_i$, and consider $(\mathbf{p}, \mathbf{q})$ to be the margins of a set of $m \times n$ matrices. First, we address the time to compute $N(\mathbf{p}, \mathbf{q})$ using Algorithm 2.2, and $M(\mathbf{p}, \mathbf{q})$ will follow easily.

Let $\mathcal{D}(\mathbf{p}, \mathbf{q})$ denote the set of nontrivial nodes $(\mathbf{u}, \bar{\mathbf{v}})$ in the directed acyclic graph (as discussed in Section 2) descending from $(\mathbf{p}, \bar{\mathbf{q}})$ (including $(\mathbf{p}, \bar{\mathbf{q}})$), where nontrivial means $(\mathbf{u}, \bar{\mathbf{v}}) \neq (\mathbf{0}, \mathbf{0})$. Let $\Delta_k(j) := \{\mathbf{s} \in \mathbb{N}^k : W(\mathbf{s}) = j\}$ for $j, k \in \mathbb{N}$. The intuitive content of the following lemma is that the graph descending from $(\mathbf{p}, \bar{\mathbf{q}})$ is contained in a union of sets $\Delta_k(j)$ with weights decreasing by steps of $p_1, \ldots, p_m$.

Lemma 6.2 (Descendants) $\mathcal{D}(\mathbf{p}, \mathbf{q}) \subset \{(\mathbf{u}, \bar{\mathbf{v}}) : \mathbf{u} = L^{j-1}\mathbf{p},\ \bar{\mathbf{v}} \in \Delta_b(t_j),\ j = 1, \ldots, m\}$, *where* $t_j = \sum_{i=j}^{m} p_i$ *and* $b = \max q_i$.

Proof By the form of the recursion, $(\mathbf{u}, \bar{\mathbf{v}}) \in \mathcal{D}(\mathbf{p}, \mathbf{q})$ if and only if for some $1 \leq j \leq m$ there exist $\mathbf{s}^1, \ldots, \mathbf{s}^{j-1}$ in $C^{\mathbf{r}^1}(p_1), \ldots, C^{\mathbf{r}^{j-1}}(p_{j-1})$ respectively, with $\mathbf{r}^1 = \bar{\mathbf{q}}$, $\mathbf{r}^{i+1} = \mathbf{r}^i \backslash \mathbf{s}^i$ for $i = 1, \ldots, j-1$, such that $(\mathbf{u}, \bar{\mathbf{v}}) = (L^{j-1}\mathbf{p}, \mathbf{r}^j)$. For $j \geq 2$, by Lemma 6.1(3 and 4),

$$W(\mathbf{r}^j) = W(\mathbf{r}^{j-1}\backslash\mathbf{s}^{j-1}) = W(\mathbf{r}^{j-1}) - p_{j-1} = W(\mathbf{r}^{j-2}) - p_{j-2} - p_{j-1}$$

$$= \cdots = W(\mathbf{r}^1) - (p_1 + \cdots + p_{j-1}) = \sum_{i=1}^{n} q_i - \sum_{i=1}^{j-1} p_i = \sum_{i=1}^{m} p_i - \sum_{i=1}^{j-1} p_i = t_j,$$

and $\mathbf{r}^j \in \mathbb{N}^b$ by construction, so $\mathbf{r}^j \in \Delta_b(t_j)$. Hence, $(\mathbf{u}, \bar{\mathbf{v}}) = (L^{j-1}\mathbf{p}, \mathbf{r}^j)$ belongs to the set as claimed. ∎

Let $T(\mathbf{p}, \mathbf{q})$ be the time (number of machine operations) required by the algorithm (Algorithm 2.2) to compute $N(\mathbf{p}, \mathbf{q})$ after precomputing all needed binomial coefficients. Let

$\tau(\mathbf{u}, \bar{\mathbf{v}})$ be the time to compute $\bar{N}(\mathbf{u}, \bar{\mathbf{v}})$ given $\bar{N}(L\mathbf{u}, \bar{\mathbf{v}} \backslash \mathbf{s})$ for all $\mathbf{s} \in C^{\bar{\mathbf{v}}}(u_1)$. That is, $T(\mathbf{p}, \mathbf{q})$ is the time to perform the entire recursive computation, whereas $\tau(\mathbf{u}, \bar{\mathbf{v}})$ is the time to perform a given call to the algorithm not including time spent in subcalls to the algorithm.

Let $n_0 := \#\{i : q_i > 0\}$ denote the number of nonempty columns. By constructing Pascal's triangle, we precompute all possible binomial coefficients that will be needed, and store them in a lookup table. We only need binomial coefficients with entries less or equal to n_0, for the following reason. In the binary case, the recursion involves numbers of the form $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ with $\mathbf{s} \leq \bar{\mathbf{v}}$, and for any descendent $(\mathbf{u}, \bar{\mathbf{v}})$ and any $i > 0$ we have $\bar{v}_i \leq n_0$ since the number of columns with sum i is less or equal to the total number of nonempty columns. For the $\mathbb{N}$-valued case, the same set of binomial coefficients will be sufficient, since then we have numbers of the form $\binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}$ with $\mathbf{s} \leq \bar{\mathbf{v}} + L\mathbf{s}$, and thus

$$\bar{v}_i + s_{i+1} \leq \bar{v}_i + \bar{v}_{i+1} + s_{i+2} \leq \cdots \leq \bar{v}_i + \bar{v}_{i+1} + \bar{v}_{i+2} + \cdots \leq n_0,$$

where the last inequality holds because the number of columns j with sum greater or equal to i is no more than the total number of nonempty columns. Since the addition of two d-digit numbers takes $\Theta(d)$ time, and there are $\binom{n_0+2}{2}$ binomial coefficients with entries less or equal to n_0, then the bound $\log \binom{j}{k} + 1 \leq n_0 \log 2 + 1$ on the number of digits for such a binomial coefficient shows that this pre-computation can be done in $\mathrm{O}(n_0^3)$ time. Except in trivial cases (when the largest column sum is 1), the additional time needed does not affect the bounds on $T(\mathbf{p}, \mathbf{q})$ that we will prove below.

We now bound the time required for a given call to the algorithm.

Lemma 6.3 (Time per call) *$\tau(\mathbf{u}, \bar{\mathbf{v}}) \leq \mathrm{O}((ab + c)(\log c)^3 |C_b(u_1)|)$ for $(\mathbf{u}, \bar{\mathbf{v}}) \in \mathcal{D}(\mathbf{p}, \mathbf{q})$, where $a = \max p_i, b = \max q_i$, and $c = \sum p_i$.*

Proof Note that we always have $\bar{v}_i \leq n_0$, since the number of columns with sum i cannot exceed the number of nonempty columns. Thus, in the recursion formula, for each $\mathbf{s}$ in the sum corresponding to $(\mathbf{u}, \bar{\mathbf{v}})$, we have the bound

$$\binom{\bar{\mathbf{v}}}{\mathbf{s}} = \prod_{i=1}^{b} \binom{\bar{v}_i}{s_i} \leq \prod_{i=1}^{b} \bar{v}_i^{s_i} \leq n_0^{\sum_i s_i} \leq n_0^a \leq c^a.$$

Let $T_m(k)$ be the time required to multiply two numbers of magnitude k or less. By the Schönhage-Strassen algorithm [30], $T_m(k) \leq \mathrm{O}((\log k)(\log\log k)(\log\log\log k))$. Therefore, $T_m(\binom{\bar{\mathbf{v}}}{\mathbf{s}}) \leq T_m(c^a) \leq \mathrm{O}(a(\log c)^3)$. Since we have precomputed the binomial coefficients, the time required to compute $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ is thus bounded by $\mathrm{O}(ab(\log c)^3)$. To finish computing the term corresponding to $\mathbf{s}$ in the recursion formula, we must multiply $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ by

$\bar{N}(L\mathbf{u}, \bar{\mathbf{v}} \backslash \mathbf{s})$. Since

$$\bar{N}(L\mathbf{u}, \bar{\mathbf{v}} \backslash \mathbf{s}) \leq N(\mathbf{p}, \mathbf{q}) \leq \prod_{i=1}^{m} \binom{n_0}{p_i} \leq \prod_{i=1}^{m} n_0^{p_i} = n_0^c \leq c^c,$$

then this multiplication can be done in $T_m(N(\mathbf{p}, \mathbf{q})) \leq T_m(c^c) \leq \mathrm{O}(c(\log c)^3)$ time. Since we are summing over $C^{\bar{\mathbf{v}}}(u_1)$, and $C^{\bar{\mathbf{v}}}(u_1) \subset C_b(u_1)$, then altogether we have $\tau(\mathbf{u}, \bar{\mathbf{v}}) \leq \mathrm{O}((ab + c)(\log c)^3 |C_b(u_1)|)$ for the time per call. ∎

Lemma 6.4 (Bound on weighted simplices) $\#\Delta_k(j) \leq \binom{j + k - 1}{k - 1}$ *for any* $j, k \in \mathbb{N}$.

Proof The map $f(\mathbf{r}) = (1r_1, 2r_2, \ldots, kr_k)$ is an injection $f : \Delta_k(j) \to C_k(j)$. Thus, $\#\Delta_k(j) \leq \#C_k(j) = \binom{j+k-1}{k-1}$. ∎

We are now ready to prove Theorem 2.3 for the case of $N(\mathbf{p}, \mathbf{q})$.

Proof of Theorem 2.3 for $N(\mathbf{p}, \mathbf{q})$

By storing intermediate results in a lookup table, once we have computed $\bar{N}(\mathbf{u}, \bar{\mathbf{v}})$ upon our first visit to node $(\mathbf{u}, \bar{\mathbf{v}})$, we can simply reuse the result for later visits. Hence, we need only expend $\tau(\mathbf{u}, \bar{\mathbf{v}})$ time for each node $(\mathbf{u}, \bar{\mathbf{v}})$ occuring in the graph. Let $t_j = \sum_{i=j}^{m} p_i$ and $d = (ab + c)(\log c)^3$. Then

$$T(\mathbf{p}, \mathbf{q}) = \sum_{(\mathbf{u}, \bar{\mathbf{v}}) \in \mathcal{D}(\mathbf{p}, \mathbf{q})} \tau(\mathbf{u}, \bar{\mathbf{v}}) \overset{\text{(a)}}{\leq} \sum_{j=1}^{m} \sum_{\bar{\mathbf{v}} \in \Delta_b(t_j)} \tau(L^{j-1}\mathbf{p}, \bar{\mathbf{v}})$$

$$\overset{\text{(b)}}{\leq} \sum_{j} \sum_{\bar{\mathbf{v}}} \mathrm{O}(d|C_b(p_j)|) = \sum_{j} \mathrm{O}(d|C_b(p_j)||\Delta_b(t_j)|)$$

$$\overset{\text{(c)}}{\leq} \sum_{j} \mathrm{O}(d \binom{p_j + b - 1}{b - 1} \binom{t_j + b - 1}{b - 1})$$

$$\overset{\text{(d)}}{\leq} \sum_{j} \mathrm{O}(d \binom{a + b - 1}{b - 1} \binom{c + b - 1}{b - 1}) \leq \mathrm{O}(dm(a + b - 1)^{b-1}(c + b - 1)^{b-1}),$$

where (a) follows by Lemma 6.2, (b) by 6.3, (c) by 6.4, and (d) since $p_j \leq a$ and $t_j \leq c$. This proves (1) and (2). Now, (3) and (4) follow from (2) since $a \leq c \leq bn$. ∎

Proof of Theorem 2.3 for $M(\mathbf{p}, \mathbf{q})$

Other than the coefficients, the only difference between the recursion for $\bar{M}(\mathbf{p}, \bar{\mathbf{q}})$ and that for $\bar{N}(\mathbf{p}, \bar{\mathbf{q}})$ is that we are summing over $\mathbf{s}$ such that $\mathbf{s} \in C^{\mathbf{r}+L\mathbf{s}}(p_1)$. Lemma 6.2 holds with the same proof, except with $C^{\mathbf{r}^1}(p_1), \ldots, C^{\mathbf{r}^{j-1}}(p_{j-1})$ replaced by $C^{\mathbf{r}^1+L\mathbf{s}^1}(p_1)$, $\ldots$, $C^{\mathbf{r}^{j-1}+L\mathbf{s}^{j-1}}(p_{j-1})$, respectively. Considering Lemma 6.3, let $(\mathbf{u}, \bar{\mathbf{v}})$ be a descendent of $(\mathbf{p}, \bar{\mathbf{q}})$ in the graph for $\bar{M}(\mathbf{p}, \bar{\mathbf{q}})$, and let $\mathbf{s}$ be such that $\mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(u_1)$. Similarly to before, recalling that $\bar{v}_i + s_{i+1} \leq n_0$ (as proven above in our discussion of precomputing the binomial coefficients), we have

$$\binom{\bar{\mathbf{v}} + L\mathbf{s}}{\mathbf{s}} = \prod_{i=1}^{b} \binom{\bar{v}_i + s_{i+1}}{s_i} \leq \prod_i (\bar{v}_i + s_{i+1})^{s_i} \leq n_0^{\sum s_i} \leq n_0^a \leq c^a.$$

This yields $T_m(\binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}) \leq T_m(c^a) \leq \mathrm{O}(a(\log c)^3)$, just as before. Since

$$\bar{M}(L\mathbf{u}, \bar{\mathbf{v}} \backslash \mathbf{s}) \leq M(\mathbf{p}, \mathbf{q}) \leq \prod_{i=1}^{m} \binom{p_i + n_0 - 1}{p_i} \leq \prod_i (2c)^{p_i} = (2c)^c,$$

then we also obtain $T_m(M(\mathbf{p}, \mathbf{q})) \leq T_m((2c)^c) \leq \mathrm{O}(c(\log c)^3)$ as before. Further, $\{\mathbf{s} : \mathbf{s} \in C^{\bar{\mathbf{v}}+L\mathbf{s}}(u_1)\} \subset C_b(u_1)$, so altogether the time per call is $\mathrm{O}((ab + c)(\log c)^3 |C_b(u_1)|)$, and thus the result of Lemma 6.3 continues to hold. With this result, the proof of the bounds goes through as well. ∎

This completes the proof of Theorem 2.3. Now, we address the time required to uniformly sample a matrix with specified margins. Let $T_r(k)$ be the maximum over $1 \leq j \leq k$ of the expected time to generate a random integer uniformly between 1 and j. If we are given a random bitstream (independent and identically distributed Bernoulli$(1/2)$ random variables) with constant cost per bit, then $T_r(k) = \mathrm{O}(\log k)$, since for any $j \leq k$, $\lceil \log_2 j \rceil \leq \lceil \log_2 k \rceil$ random bits can be used to generate an integer uniformly between 1 and $2^{\lceil \log_2 j \rceil}$ and then rejection sampling can be used to generate uniform samples over $\{1, \ldots, j\}$. Since the expected value of a Geometric(p) random variable is $1/p$, then the expected number of samples required to obtain one that falls in $\{1, \ldots, j\}$ is always less than 2. More generally, for any fixed $d \in \mathbb{N}$, if we can draw uniform samples from $\{1, \ldots, d\}$, then we have $T_r(k) = \mathrm{O}(\log k)$ by considering the base-d analogue of the preceding argument.

Lemma 6.5 (Sampling time) *Algorithm 2.4 takes* $\mathrm{O}(mT_r(n^c) + maT_r(n) + mb \log(a+b))$ *expected time per sample in the binary case, and* $\mathrm{O}(mT_r((2c)^c) + maT_r(n) + mb \log(a + b))$ *expected time per sample in the* $\mathbb{N}$*-valued case. If* $T_r(k) = \mathrm{O}(\log k)$*, then this is* $\mathrm{O}(mc \log c)$ *expected time per sample in both cases.*

Remark If b is bounded then $\mathrm{O}(mc \log c) \leq \mathrm{O}(mn \log n)$ since $c \leq bn$, and so this is polynomial expected time for bounded column sums.

Proof By the form of the recursion, the depth of the graph descending from $(\mathbf{p}, \bar{\mathbf{q}})$ is equal to the number of rows m, since $\mathbf{p} \in \mathbb{N}^m$ and thus $L^m\mathbf{p} = \mathbf{0}$. For each of the m iterations of the sampling algorithm, we begin at some node $(\mathbf{u}, \bar{\mathbf{v}})$, and we must (A) randomly choose a child $(L\mathbf{u}, \bar{\mathbf{v}}\backslash\mathbf{s})$ with probability proportional to its count times the number of corresponding rows, and then (B) choose a row uniformly from among the $\binom{\bar{\mathbf{v}}}{\mathbf{s}}$ possible choices in the binary case (or $\binom{\bar{\mathbf{v}}+L\mathbf{s}}{\mathbf{s}}$ in the $\mathbb{N}$-valued case).

First consider the binary case. To randomly choose a child, consider a partition of the integers $1, \ldots, N(\mathbf{u}, \mathbf{v})$ with each part corresponding to a term in the recursion formula for $\bar{N}(\mathbf{u}, \bar{\mathbf{v}})$. Generate an integer uniformly at random between 1 and $N(\mathbf{u}, \mathbf{v})$, and choose the corresponding child. Generating such a random number takes $T_r(N(\mathbf{u}, \mathbf{v})) \leq T_r(N(\mathbf{p}, \mathbf{q})) \leq T_r(n_0^c)$ time. Since there are no more than $\binom{a+b-1}{b-1} \leq (a+b-1)^{b-1}$ children at any step, one can determine which child corresponds to the chosen number in $\mathrm{O}((b-1)\log(a+b-1))$ time by organizing the children in a binary tree. So (A) takes $\mathrm{O}(T_r(n_0^c) + b\log(a+b))$ time. Choosing a row consists of uniformly sampling a subset of size s_i from a set of $\bar{v}_i$ elements, for $i = 1, \ldots, b$. Sampling such a subset can be done by sampling without replacement s_i times, which takes $\sum_{j=0}^{s_i-1} T_r(\bar{v}_i - j) \leq s_i T_r(n_0)$ time. So (B) can be done in $\sum_{i=1}^{b} s_i T_r(n_0) \leq aT_r(n_0)$ time. Repeating this process m times, once for each row, we see that sampling a matrix takes $\mathrm{O}(mT_r(n_0^c) + mb\log(a+b) + maT_r(n_0))$ time. If $T_r(k) \leq \mathrm{O}(\log k)$, this is $\mathrm{O}(mc\log n_0 + mb\log(a+b) + ma\log n_0) \leq \mathrm{O}(mc\log c)$ since $a, b, n_0 \leq c$.

For the $\mathbb{N}$-valued case, the same argument applies, replacing $N(\mathbf{u}, \mathbf{v})$ with $M(\mathbf{u}, \mathbf{v})$, n_0^c with $(2c)^c$, and $\bar{v}_i$ with $\bar{v}_i + s_{i+1}$. ∎

References

[1] Harsh Anand, Vishwa Chander Dumir, and Hansraj Gupta, *A combinatorial distribution problem*, Duke Mathematical Journal **33** (1966), no. 4, 757–769.

[2] Alexander I. Barvinok, *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Mathematics of Operations Research **19** (1994), no. 4, 769–779.

[3] Matthias Beck and Dennis Pixton, *The Ehrhart polynomial of the Birkhoff polytope*, Discrete and Computational Geometry **30** (2003), no. 4, 623–637.

[4] E. Canfield, C. Greenhill, and B. McKay, *Asymptotic enumeration of dense 0-1 matrices with specified line sums*, Journal of Combinatorial Theory, Series A **115** (2008), no. 1, 32–66.

[5] E. Rodney Canfield and Brendan D. McKay, *Asymptotic enumeration of dense 0-1 matrices with equal row sums and equal column sums*, The Electronic Journal of Combinatorics **12** (2005), no. 2, R29.

[6] T. J. Case and M. L. Cody, *The land birds*, Island Biogeography in the Sea of Cortez, University of California Press, Berkeley, CA, 1983, pp. 210–245.

[7] Yuguo Chen, Persi Diaconis, Susan P. Holmes, and Jun S. Liu, *Sequential Monte Carlo methods for statistical analysis of tables*, Journal of the American Statistical Association **100** (2005).

[8] P. Diaconis and A. Gangolli, *Rectangular arrays with fixed margins*, Discrete Probability and Algorithms, Springer-Verlag, New York, 1995, pp. 15–41.

[9] Persi Diaconis and Bradley Efron, *Testing for independence in a two-way table: New interpretations of the chi-square statistic*, Annals of Statistics **13** (1985), no. 3, 845–874.

[10] M. Dyer, R. Kannan, and J. Mount, *Sampling contingency tables*, Random Structures and Algorithms **10** (1997), 487–506.

[11] D. Gale, *A theorem on flows in networks*, Pacific Journal of Mathematics **7** (1957), 1073–1082.

[12] Ira M. Gessel, *Enumerative applications of symmetric functions*, Séminaire Lotharingien de Combinatoire **B17a** (1987), 5–21.

[13] _____, *Symmetric functions and P-recursiveness*, Journal of Combinatorial Theory, Series A **53** (1990), no. 2, 257–285.

[14] C. Greenhill, B. McKay, and X. Wang, *Asymptotic enumeration of sparse 0-1 matrices with irregular row and column sums*, Journal of Combinatorial Theory, Series A **113** (2006), no. 2, 291–324.

[15] J. Hammond, *On the calculation of symmetric functions*, Proceedings of the London Mathematical Society **XIII** (1882), 79.

[16] Matthew T. Harrison, *A dynamic programming approach for approximate uniform generation of binary matrices with specified margins, arXiv:0906.1004 [stat.CO]*, 2009.

[17] R. B. Holmes and L. K. Jones, *On uniform generation of two-way tables with fixed margins and the conditional volume test of Diaconis and Efron*, The Annals of Statistics **24** (1996), no. 1, 64–68.

[18] Ben Johnsen and Eldar Straume, *Counting binary matrices with given row and column sums*, Mathematics of Computation **48** (1987), no. 178, 737–750.

[19] Jesús A. De Loera, Raymond Hemmecke, Jeremiah Tauzer, and Ruriko Yoshida, *Effective lattice point counting in rational convex polytopes*, Journal of Symbolic Computation **38** (2004), no. 4, 1273–1302.

[20] Jesús A. De Loera and Bernd Sturmfels, *Algebraic unimodular counting*, Mathematical Programming **96** (2003), no. 2, 183–203.

[21] Percy A. MacMahon, *Combinatory analysis*, vol. I,II, Cambridge University Press, London, 1915.

[22] Brendan D. McKay, *Applications of a technique for labeled enumeration*, Congressus Numerantium **40** (1983), 207–221.

[23] John Mount, *Fast unimodular counting*, Combinatorics, Probability, and Computing **9** (2000), no. 3.

[24] Blanca Rosa Pérez-Salvador, Sergio de-los Cobos-Silva, Miguel Angel Gutiérrez-Ándrade, and Adolfo Torres-Chazaro, *A reduced formula for the precise number of (0,1)-matrices in A(R,S)*, Discrete Mathematics **256** (2002), no. 1-2, 361–372.

[25] Dennis M. Power, *Numbers of bird species on the California Islands*, Evolution **26** (1972), no. 3, 451–463.

[26] R. C. Read, *The enumeration of locally restricted graphs (I)*, Journal of the London Mathematical Society **s1-34** (1959), no. 4, 417–436.

[27] ———, *The enumeration of locally restricted graphs (II)*, Journal of the London Mathematical Society **s1-35** (1960), no. 3, 344–351.

[28] J. Howard Redfield, *The theory of group-reduced distributions*, American Journal of Mathematics **49** (1927), no. 3, 433–455.

[29] H. Ryser, *Combinatorial properties of matrices of zeros and ones*, Canadian Journal of Mathematics **9** (1957), 371–377.

[30] A. Schönhage and V. Strassen, *Schnelle multiplikation großer zahlen*, Computing **7** (1971), no. 3-4, 281–292.

[31] Richard P. Stanley, *Linear homogeneous Diophantine equations and magic labelings of graphs*, Duke Mathematical Journal **40** (1973), no. 3, 607–632.

[32] Bo-Ying Wang, *Precise number of (0,1)-matrices in U(R,S)*, Scientia Sinica, Series A **XXXI** (1988), no. 1, 1–6.

[33] Bo-Ying Wang and Fuzhen Zhang, *On the precise number of (0,1)-matrices in U(R,S)*, Discrete Mathematics **187** (1998), 211–220.

A Enumeration results

Binary matrices with margins $(70, 30, 20, 10, 5^{(6)}, 4^{(10)}, 3^{(20)}, 2^{(60)}), (4^{(80)}, 3^{(20)})$

860585058801817078819959949756041558231879514104670757612387
280341919502865086909993523205599348663646837362726765460951
032776118129432733489342067673016169716787054236343091407458
802261593735765113169808512677339861494709092492858489355535
514748397544147637928475318462070009855280569561693514768239
201499080842592443823774161366680107327323365049702068246736
456919918589686056321467354298509024976141650428747522863473
529515269318246400000000000000000000000

$\mathbb{N}$-valued matrices with margins $(70, 30, 20, 10, 5^{(6)}, 4^{(10)}, 3^{(20)}, 2^{(60)}), (4^{(80)}, 3^{(20)})$

620017488391049592297896956531192562528805388295441812965295
130897484012791595142882674755488640101825726867156331426482
441148514978852842582445295040041143220637964258279947442682
896809706562683189375098411751981435132377208717294759756041
358372207736032818841045369779439398975681041714752821787419
816573563436066161167632677774184809010338787868042742993719
703936093873250600121874335524794990013547042810153560084573
133035731217642637607153615611029851392000000000000000000000
000

Ehrhart polynomials $H_n(r)$ **for** $n = 4, \ldots, 8$

$H_4(1) = 24$
$H_4(2) = 282$
$H_4(3) = 2008$

$H_5(1) = 120$
$H_5(2) = 6210$
$H_5(3) = 153040$
$H_5(4) = 2224955$
$H_5(5) = 22069251$

$H_5(6) = 164176640$

$H_6(1) = 720$
$H_6(2) = 202410$
$H_6(3) = 20933840$
$H_6(4) = 1047649905$
$H_6(5) = 30767936616$
$H_6(6) = 602351808741$
$H_6(7) = 8575979362560$
$H_6(8) = 94459713879600$
$H_6(9) = 842286559093240$
$H_6(10) = 6292583664553881$

$H_7(1) = 5040$
$H_7(2) = 9135630$
$H_7(3) = 4662857360$
$H_7(4) = 936670590450$
$H_7(5) = 94161778046406$
$H_7(6) = 5562418293759978$
$H_7(7) = 215717608046511873$
$H_7(8) = 5945968652327831925$
$H_7(9) = 123538613356253145400$
$H_7(10) = 2023270039486328373811$
$H_7(11) = 27046306550096288483238$
$H_7(12) = 303378141987182515342992$
$H_7(13) = 29200543364925217205722 76$
$H_7(14) = 24563127009195223721952590$
$H_7(15) = 183343273080700916973016745$

$H_8(1) = 40320$
$H_8(2) = 545007960$
$H_8(3) = 1579060246400$
$H_8(4) = 1455918295922650$
$H_8(5) = 569304690994400256$
$H_8(6) = 114601242382721619224$
$H_8(7) = 13590707419428422843904$
$H_8(8) = 1046591482728407939338275$
$H_8(9) = 562727224063492350359168 00$
$H_8(10) = 22331603423698255967021487 20$
$H_8(11) = 68316292103293669997188919040$
$H_8(12) = 16679320988627738377348230 42196$
$H_8(13) = 33427469280977307618866364694400$
$H_8(14) = 56279880567334201675236634 4185200$

$H_8(15) = 81152089774654048741002264925753 60$
$H_8(16) = 10185706615053029414642861591795 7029$
$H_8(17) = 11282825264050225540495573290972 52992$
$H_8(18) = 11161302946841260178530673680176000200$
$H_8(19) = 99613494890126594335550124219924540800$
$H_8(20) = 80925677061054067545465751719401868 0846$
$H_8(21) = 60311079898755627512661169019993277 10720$