# Approximately Counting Integral Flows and Cell-Bounded Contingency Tables[*]

Mary Cryan[†]
School of Informatics
University of Edinburgh
Edinburgh EH9 3JZ, UK.
mcryan@inf.ed.ac.uk

Martin Dyer[‡]
School of Computing
University of Leeds
Leeds LS2 9JT, UK.
dyer@comp.leeds.ac.uk

Dana Randall[§]
College of Computing
Georgia Inst. of Technology
Atlanta GA 30332-0280.
randall@cc.gatech.edu

## ABSTRACT

We consider the problem of approximately counting integral flows in a network. We show that there is an *fpras* based on volume estimation if all capacities are sufficiently large, generalising a result of Dyer, Kannan and Mount (1997). We apply this to approximating the number of contingency tables with prescribed *cell bounds* when the number of rows is constant, but the row sums, column sums and cell bounds may be arbitrary. We provide an *fpras* for this problem via a combination of dynamic programming and volume estimation. This generalises an algorithm of Cryan and Dyer (2002) for standard contingency tables, but the analysis here is considerably more intricate.

## Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

## General Terms

Algorithms, Theory

## Keywords

Approximate counting, contingency tables, integral flows

## 1. INTRODUCTION

In this paper we consider two related counting problems. First we consider the problem of counting *integral flows in a general capacitated network*. A special case of this problem was considered by Kannan and Vempala [20]. They gave an fpras (fully polynomial randomized approximation scheme) to approximately count integral $s-t$ flows in an undirected network when all edge capacities are sufficiently large. Recently Baldoni-Silva, De Loera and Vergne showed that integer-valued flows in a general capacitated network can be represented as lattice points inside a related *flow polytope* [3]. Hence they construct exact counting algorithms using Barvinok's approach [2] to counting lattice points in a fixed-dimensional polytope. Their algorithms run in polynomial-time if the dimension of the *flow polytope* is constant. Some applications are also discussed in [3].In general, exactly counting lattice points is #P-Complete [15, 27], and only approximation is possible in polynomial-time. Jerrum, Sinclair and Vigoda [18] gave an *fpras* for the special case of *0-1 flows*, where all capacities are 0 or 1. By contrast, our first contribution in this paper (in §2) is to show there is an *fpras*, based on sampling and volume estimation for convex bodies [13], whenever the minimum (tight) capacity in the network (as defined in §2 below) is sufficiently large. Interestingly, the proof relies on the properties of the *maximum spanning tree* in the network (using the capacities as weights) to show that the flow polytope is *well-rounded* [16]. We note that establishing this property is far from straightforward for general flow polytopes, whereas it follows directly for the special case considered in [20].

The result of §2 can be applied directly to counting *cell-bounded contingency tables*, which we now define: Let $[\ell]$ denote the set $\{1,\ldots,\ell\}$. Suppose we are given a list of positive integers $r=(r_1,\ldots,r_m)$ called the *row sums*, another list of positive integers $c=(c_1,\ldots,c_n)$ called the *column sums*, and a *cell bound* $b_{ij}$, for every $i\in[m], j\in[n]$. Assume that $b_{ij}\le\min\{r_i,c_j\}$ for all $i\in[m], j\in[n]$, and that $\sum_{i=1}^m r_i=\sum_{j=1}^n c_j$ (the *table sum*). Then the set $\Sigma_{r,c,b}$ of *cell-bounded contingency tables* is the set of all $m\times n$ non-negative integer matrices x which satisfy the row and column sums, and also satisfy $x_{ij}\le b_{ij}$ for all $i\in[m], j\in[n]$. We will see in §3 that cell-bounded contingency tables are equivalent to integer flows in an appropriately-defined bipartite network. The definition of $\Sigma_{r,c,b}$ is a generalisation of the set $\Sigma_{r,c}$ of *standard* contingency tables (where the cell bounds are $b_{ij}=\min\{r_i,c_j\}$ for all $i\in[m], j\in[n]$).

The problem of sampling standard contingency tables from $\Sigma_{r,c}$ almost uniformly at random has important applications in practical statistics (see Diaconis and Efron [7]). Much

[*]This work was partly done while the authors were visiting the Isaac Newton Institute for Mathematical Sciences, Cambridge, UK.

[†]Supported by EPSRC grant GR/S76175/01.

[‡]Supported by EPSRC grant GR/S76151/01.

[§]Supported in part by NSF grant CCR-0105639.

work has been carried out on this and on the related problem of approximating the total number of tables $|\Sigma_{r,c}|$ (see, e.g., [4, 5, 6, 8, 9, 12, 15, 17, 23, 24]). The algorithm of Barvinok [2] for counting lattice points in a polytope can be used to count contingency tables *exactly* in polynomial-time when the numbers of rows and columns are both constant. Dyer, Kannan and Mount [15] showed that, whenever the row sums satisfy $r_i \in \Omega(n^2m)$ for all $i \in [m]$ and the column sums satisfy $c_j \in \Omega(m^2n)$ for all $j \in [n]$, there is an *fpras* for counting standard contingency tables. This was improved later by Morris [22]. Dyer and Greenhill [12] gave an *fpras* to approximately count contingency tables with two rows by proving a natural Markov chain called the $2 \times 2$ heat bath chain is rapidly mixing. Subsequently, Cryan and Dyer [4] gave an *fpras* based on a combination of dynamic programming and volume estimation to count tables when the number of rows $m$ is a constant. Cryan et al. [5] gave an alternative algorithm by proving that the $2 \times 2$ heat bath Markov chain on such tables is rapidly mixing. Later, Dyer [11] gave an *fpras* based entirely on dynamic programming, improving substantially on the running time of the algorithms in [4] and [5]. However, the existence of an *fpras* for counting standard contingency tables in the general case remains a notorious open problem.

Counting and sampling cell-bounded contingency tables also has natural applications in statistics, in the case where each cell has a known maximum. According to Diaconis and Gangolli [8], this is a "practical class of problems". However, there appears to have been little work on this problem except in the case where some cell upper bounds may be zero (so-called *structural zeros*). The Markov chain used in [5, 12], which updates values in a $2 \times 2$ submatrix during each step of the simulation, is no longer viable in the cell-bounded case since the state space might not even be connected; for example, a $3 \times 3$ table with structural zeros on the diagonal has no allowable moves. Aoki [1] and Rapollo [25] recently considered the design of alternative chains for sampling such tables using the "Markov basis" approach of Diaconis and Sturmfels [10]. The required sampling distribution will be uniform when the conditional volume test of [7] is employed. However, the main focus of these papers is on sampling from the hypergeometric distribution, which is easier than the general case. These Markov chains take small steps, so cannot lead to polynomial time sampling unless all numbers are given in unary.

There are general theoretical reasons for studying cell-bounded tables. It is well-known that for any *self-reducible* relation, the problem of obtaining an *fpras* for approximate counting is equivalent to the problem of finding an *fpaus* (fully polynomial almost uniform sampler) [19]. Standard contingency tables are not known to be self-reducible, which is unusual. However there is a simple self-reducible characterisation of cell-bounded contingency tables. Another interesting fact about cell-bounded contingency tables is that they generalise the concept of perfect matchings in a bipartite graph (where all cell-bounds are 0 or 1). Approximating the number of perfect matchings in a bipartite graph ( the *0-1 permanent*) was an important open problem for several years, until Jerrum, Sinclair and Vigoda [18] finally established the existence of an *fpras*.

The second overall contribution of our paper is to put the problem of approximately counting *cell-bounded* contingency tables on precisely the same footing as that of approximately counting *standard* contingency tables. One immediate consequence of the main result of §2 is that, if all cell bounds are "sufficiently large" (in a sense we define later), then we can use sampling and volume estimation for the flow polytope to obtain an *fpras* to count approximately the number of cell-bounded tables. This result does not depend on the number of rows or columns. It can be seen as a direct generalisation to the cell-bounded case of the result of Dyer, Kannan and Mount [15] for standard contingency tables.

Following the other thread of results for contingency tables, in §3 we assume that the number of rows is constant but make no assumptions about the size of the cell bounds. We show that we can combine dynamic programming and volume estimation to design an *fpras* in this case. This *fpras* is broadly similar to that of Cryan and Dyer [4], but the structure of cell-bounded tables is considerably more intricate. However, since cell-bounded contingency tables may be viewed as integral flows in a bipartite network, we are able to extend the approach of §2. Our proof relies even more strongly on properties of the maximum spanning tree.

These are the first results that provide provably efficient counting and sampling algorithms for non-trivial classes of cell-bounded contingency tables. Moreover, they demonstrate that these seemingly broader classes of problems might not be harder than counting and sampling standard contingency tables without cell bounds. The question of existence of an *fpras* for the general problem of counting cell-bounded contingency tables remains open, as it does for standard contingency tables. However, since an *fpras* for general cell-bounded tables would include approximating the 0-1 permanent [18] as a special case, it may prove elusive. On the other hand, the results of this paper are some indication that standard contingency tables have no exploitable structure beyond that which exists in the cell-bounded case. Consequently, there is no obvious reason to expect that an *fpras* for standard contingency tables may be found any more easily than one for the general cell-bounded case.

## 2. COUNTING INTEGRAL FLOWS

Suppose that we have a flow network $\mathcal{N} = (V, A)$ with *capacities* $b(a) \in \mathbb{Z}^+ \cup \{\infty\}$ $(a \in A)$ and *supplies* $\rho(v) \in \mathbb{Z}$ $(v \in V)$ such that $\sum_{v \in V} \rho(v) = 0$. We will write $n = |V|$, $m = |A|$ and $d = m - n + 1$. A flow $x$ must satisfy

$$\sum_{w:a=(v,w)} x(a) - \sum_{w:a=(w,v)} x(a) = \rho(v) \qquad (v \in V), \quad (1)$$

$$0 \leq x(a) \leq b(a) \qquad (a \in A). \quad (2)$$

The solution set of (1)–(2), the set of *flows*, is a convex polyhedron. We are interested in estimating the number of integral flows. Any capacitated flow problem can be put in the form (1)–(2) without changing the size of the solution set. We allow $\mathcal{N}$ to contain parallel and antiparallel arcs between any two vertices since replacing these with a single edge might alter the number of integer flows satisfying (1)-(2). We may dispose of the case in which the number of solutions is infinite, corresponding to the existence of a directed cycle in the set $\{a \in A : b(a) = \infty\}$. If no such cycle exists, the system (1)–(2) determines a *flow polytope* $\mathcal{P}$. The integer solutions are the *lattice points* inside $\mathcal{P}$, and we use $\mathcal{I}(\mathcal{P})$ to denote the set of all such lattice points.

Our goal in this section is to develop an *fpras* for counting the number of integral flows when the minimum capacity is sufficiently large. That is, given an error tolerance $\epsilon \in (0, \frac{1}{2})$, we will design an algorithm that runs in time polynomial in $n, \epsilon^{-1}$, and $\log(\max_a b(a))$ and produces an estimate $|\mathcal{I}'|$ of $|\mathcal{I}(\mathcal{P})|$ that satisfies

$$(1-\epsilon)|\mathcal{I}(\mathcal{P})| \leq |\mathcal{I}'| \leq (1+\epsilon)|\mathcal{I}(\mathcal{P})|$$

with high probability. We accomplish this by relating $|\mathcal{I}(\mathcal{P})|$ to the volume of $\mathcal{P}$.

We assume that $\mathcal{N}$ has *tight capacities*, defined as follows.

Definition 1. *A network $\mathcal{N}$ has* tight capacities *if for each $a \in A$, there exist flows $f_a^-$, $f_a^+$ satisfying $f_a^-(a) = 0$ and $f_a^+(a) = b(a)$ (and $b(a) > 0$).*

If the network does not satisfy this condition, we can make a polynomial-time transformation to define a new network with the same number of integral flows. For each arc $a$, we find $\max_x x(a)$, the maximum value of $x(a)$ over all flows, by solving a minimum cost flow problem. (See, for example, Schrijver [26, §12]). Similarly, for every $a \in A$, we find $\min_x x(a)$. We let the new capacities be $\big(\max_x x(a) - \min_x x(a)\big)$ $(a \in A)$, and let the new supplies be

$$\rho(u) - \sum_{(u,v)\in A} \min_x x(u,v) + \sum_{(w,u)\in A} \min_x x(w,u) \qquad (\forall u \in V).$$

We also may assume that $b(a) > 0$ for all $a \in A$, since otherwise the arc $a$ can be deleted from $A$.

Using the convexity of the flow polytope, we can now define an *internal flow*.

Definition 2. *Given a flow polytope $\mathcal{P}$ with tight capacities, we define an* internal flow

$$f =_{def} \frac{1}{2m} \sum_{a\in A} (f_a^- + f_a^+) \qquad (3)$$

In general $f$ will not be an integral flow. We will use *rational* flow for functions like $f$ whereas, without qualification, flow will mean *integral* flow. Note that $\frac{1}{2m} b(a) \leq f(a) \leq b(a) - \frac{1}{2m} b(a)$ for all $a \in A$, using the tightness of the capacities. For any flow $x$, define the *slack* for $x$ on arc $a$ as $\min\{x(a), b(a) - x(a)\}$. Thus our internal flow $f$ has slack at least $b(a)/2m$ on every arc $a$.

We now define the concept of a *maximum spanning tree* of a network. We may assume $\mathcal{N}$ is connected, since otherwise we can consider each component separately and take the product of the number of solutions for each component.

Definition 3. *Consider the (connected) undirected multigraph $G = (V, E)$ underlying $\mathcal{N}$. We will abuse notation and refer to an* arc $a \in A$ *as an* edge $a \in E$*, forgetting its direction. Let $E$ have edge weights $b(a)$. Then a* maximum spanning tree *for $\mathcal{N}$ is any* maximum weight spanning tree *$T$ in $G$.*

It follows from standard network flow theory [26, §13] that we can eliminate the variables $x(a)$ $(a \in T)$ from the system of equations (1)–(2) to give a system of $2m$ inequalities in $d = (m - n + 1)$ bounded variables $x(a)$ $(a \in A' = A \setminus T)$.

We now show, using the spanning tree $T$, that there is an ellipsoid, and also a ball, centred at $f$, lying entirely inside $\mathcal{P}$. The approach of using a spanning tree of a network to define a full-dimensional representation of the flow polytope has been used before [20], but it is the idea of using the *maximum-weight* spanning tree which drives our result.

Theorem 1. *Let $\mathcal{N} = (V, A)$ be a connected network with tight capacities $b(\cdot)$. Let $T$ be any* maximum spanning tree *for $\mathcal{N}$. Let $b_{min} = \min_{a\in A\setminus T} b(a)$, and let $\delta = b_{min}/2m\sqrt{d}$ (where $d = m - n + 1$). Then the flow polytope $\mathcal{P}$ contains the ball $\mathcal{B}(f, \delta)$.*

Proof. For any $a \in A' = A \setminus T$, consider the "rational flow" $g_a^+$ defined by

$$\begin{aligned} g_a^+(a) &= f(a) + \tfrac{1}{2m} b(a) && \text{and} \\ g_a^+(a') &= f(a') && (a' \neq a, a' \in A'). \end{aligned}$$

Clearly $0 \leq g^+(a') \leq b(a')$ for all $a' \in A'$. To define a rational flow, we must complete this with feasible values of $g_a^+(a')$ $(a' \in T)$. First, we follow the unique circuit $C_a \subseteq T \cup \{a\}$ in the direction of $a$. For each edge $a' \neq a$ traversed, if the direction of $a'$ is the same as that of $a$, let $g_a^+(a') = f(a') + b(a)/2m$; alternatively, if $a'$ has the opposite direction to $a$, $g_a^+(a') = f(a') - b(a)/2m$. We set $g_a^+(a') = f(a')$ for all $a' \in T \setminus C_a$. This ensures that $g_a^+$ still satisfies all the supplies $\rho(.)$. It follows that for all $a' \in C_a$ we have $b(a) \leq b(a')$ and

$$g_a^+(a') \in f(a') \pm \tfrac{1}{2m} b(a) \in f(a') \pm \tfrac{1}{2m} b(a') \in [0, b(a')]$$

since $T$ is a *maximum* spanning tree. Similarly we can find $g_a^-$ such that $g_a^-(a) = f(a) - \frac{1}{2m} b(a)$ and $g_a^-(a') = f(a')$ $(a' \neq a, a' \in A')$. Now consider $\mathcal{P}$ as a polytope in $\mathbb{R}^d = \mathbb{R}^{m-n+1}$, determined by $x(a)$ $(a \in A')$. From the properties of $f$, $\mathcal{P}$ is contained in the hyper-rectangle

$$-(1 - \tfrac{1}{2m}) b(a) \leq x(a) - f(a) \leq (1 - \tfrac{1}{2m}) b(a) \qquad (a \in A').$$

Thus $\mathcal{P}$ is contained in the ellipsoid

$$\mathcal{E}_{\text{ext}} = \left\{ x : \sum_{a\in A'} \left( \frac{x(a) - f(a)}{b(a)} \right)^2 \leq d\big(1 - \tfrac{1}{2m}\big)^2 \right\}.$$

We have shown that $\mathcal{P}$ contains $\mathcal{H} = \text{conv}\{g_a^+, g_a^- : a \in A'\}$. This is an axis-scaled $\ell_1$-ball, so

$$\mathcal{H} = \left\{ x : \sum_{a\in A'} \left| \frac{x(a) - f(a)}{b(a)} \right| \leq \frac{1}{2m} \right\}.$$

Now, using Cauchy-Schwarz, $\mathcal{H}$ contains the ellipsoid

$$\mathcal{E}_{\text{int}} = \left\{ x : \sum_{a\in A'} \left( \frac{x(a) - f(a)}{b(a)} \right)^2 \leq \frac{1}{4m^2 d} \right\}.$$

which is $\mathcal{E}_{\text{ext}}$ scaled by $d(2m-1)$. Let $b_{\min} = \min_{a\in A'} b(a)$ and $\delta = b_{\min}/2m\sqrt{d}$. Then $\mathcal{E}_{\text{int}}$ contains the ball $\mathcal{B}(f, \delta)$. □

We now show $|\mathcal{I}(\mathcal{P})|$ is close to $\text{vol}(\mathcal{P})$ when $b_{\min}$ is sufficiently large.

Theorem 2. *Let $\mathcal{N} = (V, A)$ be a connected network with tight capacities $b(\cdot)$. Let $T$ be a maximum spanning tree for $\mathcal{N}$. Let $\mathcal{P}$ be the flow polytope in $\mathbb{R}^d$, with axes indexed by $A \setminus T$, where $d = m - n + 1$. If $b_{\min} > 2md$ and $\epsilon \geq 2md^2/(b_{\min} - 2md)$, then*

$$e^{-\epsilon} \text{vol}(\mathcal{P}) \quad \leq \quad |\mathcal{I}(\mathcal{P})| \quad \leq \quad e^{\epsilon} \text{vol}(\mathcal{P}).$$

Proof. Our proof is similar in principle to that of Theorem 3 in [4], though the polytope we consider here is more constrained. We use a full-dimensional representation of $\mathcal{P}$ in $\mathbb{R}^d$, relative to the rational flow $f$ defined in (3). Thus, we translate $z \in \mathbb{R}^d$ to $z'$ so that $z'(a') = z(a') - f(a)$, for $a' \in A' = A \setminus T$. Then, for each lattice point $z' = z - f \in \mathcal{I}(\mathcal{P})$, we associate with $z'$ the hypercube $H(z)$ in $d$-dimensions, where $y \in H(z)$ iff $z'(a) \le y(a) < z'(a) + 1$ for all $a \in A'$. Let $\mathcal{C} = \cup_{z \in \mathcal{I}(\mathcal{P})} H(z)$. Clearly $|\mathcal{I}(\mathcal{P})| = \mathrm{vol}(\mathcal{C})$. We will refer to the *dilation* $\alpha\mathcal{Q}$ of a $d$-dimensional convex polytope $\mathcal{Q}$ as $\{\alpha x : x \in \mathcal{Q}\}$. It is well-known that this has volume $\mathrm{vol}(\alpha\mathcal{Q}) = \alpha^d \mathrm{vol}(\mathcal{Q})$. The theorem is proven in two parts.

We show that $\mathcal{C} \subseteq (1 + \epsilon/d)\mathcal{P}$. For every $z \in \mathcal{I}(\mathcal{P})$ and every $y \in H(z)$, $\mathrm{dist}(y, z) \le \sqrt{d}$, where $\mathrm{dist}(\cdot, \cdot)$ denotes Euclidean distance. Suppose $y \notin \mathcal{P}$. Clearly $\mathrm{dist}(y, \mathcal{P}) \le \sqrt{d}$. Also, we know that there is a ball of radius $\delta$ with centre $f$ lying inside $\mathcal{P}$. The dilation $(1+\epsilon/d)\mathcal{P}$ will therefore contain all points with distance at most

$$\frac{\epsilon\delta}{d} \ge \frac{b_{\min}\sqrt{d}}{(b_{\min} - 2md)} > \sqrt{d}$$

from $\mathcal{P}$. So $H(z) \subseteq (1 + \epsilon/d)\mathcal{P}$ for every $z \in \mathcal{I}(\mathcal{P})$.

In a similar way, we can show that $(1 + \epsilon/2d)^{-1}\mathcal{P} \subseteq \mathcal{C}$. Observing that $(1 + \epsilon/d)^d \le e^\epsilon$ completes the proof. $\square$

COROLLARY 3. *Let $\mathcal{N} = (V, A)$ be a connected network with tight capacities and let $\mathcal{P}$ be the corresponding flow polytope in $\mathbb{R}^d$. If $b_{\min} \in \Omega(md^2/\log m)$ then there is an fpras for the number of integral flows $|\mathcal{I}(\mathcal{P})|$.*

PROOF. To use volume estimation to design an *fpras*, Theorem 2 is useful only if $e^\epsilon$ is polynomially bounded in $m$ and $d$. Since $m \ge d$, this is guaranteed when $b_{\min} \in \Omega(md^2/\log m)$, with the approximation error depending on $\epsilon$. We can further improve this to get relative error at most $\eta > 0$ as follows. First we determine $\mathrm{vol}(\mathcal{P})$ to relative error $\eta/3$, with high probability, using an *fpras* for volume estimation (see, e.g., [13, 21]). The original volume algorithm is due to Dyer, Frieze and Kannan [13], but many improvements now exist, the most recent being that of Lovász and Vempala [21]. Similarly, we can sample a point $x$ from $\mathcal{P}$ using the *fpaus* of [13], or one of its successors. Suppose we sample $x \in \mathcal{P}$ almost uniformly in this way. Let $\mathcal{P}' = (1 + \epsilon/d)\mathcal{P}$, so $x' = (1 + \epsilon/d)x$ is a uniform sample from $\mathcal{P}'$. Note that $\mathrm{vol}(\mathcal{P}') = (1 + \epsilon/d)^d \mathrm{vol}(\mathcal{P})$. Now $|\mathcal{I}(\mathcal{P})| = \mathrm{vol}(\mathcal{C})$, and $\mathcal{C} \subseteq \mathcal{P}'$. Let $p = \mathrm{vol}(\mathcal{C})/\mathrm{vol}(\mathcal{P}')$. We have $p \ge e^{-2\epsilon}$ from Theorem 2, so by assumption it is bounded below by an inverse polynomial. Note we can recognise when $x' \in \mathcal{C}$ by rounding down to find the unique $y \in \mathbb{Z}^d$ such that $x' \in H(y)$ and checking whether $y \in \mathcal{P}$. Thus, with high probability, we can obtain an approximation $\hat{p}$ for $p$ with relative error at most $\eta/3$ by taking $O(e^{2\epsilon}/\eta^2)$ samples. We can estimate $\mathrm{vol}(\mathcal{C})$ as $\hat{p}(1+\epsilon/d)^d \mathrm{vol}(\mathcal{P})$. Now the relative error is at most $\eta$, so this procedure gives an *fpras* (and an *fpaus*) for any class of flow problems satisfying $b_{\min} \in \Omega(md^2/\log m)$. $\square$

Kannan and Vempala [20] have given an improvement, using randomized rounding, on the above method for approximating the number of lattice points in a polytope, which was first used in [14]. They consider a polytope $\mathcal{Q}$ of dimension $d$ with $k$ facets and show that, if $\mathcal{Q}$ contains a ball of radius $\Omega(d\sqrt{\log k})$, then there is an fpras for $\mathcal{I}(\mathcal{Q})$. There are at most $2m$ facets of the flow polytope $\mathcal{P}$, so we can employ Kannan and Vempala's result if we have a ball of radius $\Omega(d\sqrt{\log 2m})$ inside $\mathcal{P}$. By Theorem 1, this gives an fpras for the number of integral flows if $b_{\min} = \Omega(d^{3/2}m\sqrt{\log m})$, an improvement on Corollary 3.

## 3. CONTINGENCY TABLES WITH CELL BOUNDS

We now consider cell-bounded contingency tables with $m$ rows and $n$ columns, row sums $r_i > 0$ ($i \in [m]$) and column sums $c_j > 0$ ($j \in [n]$). The upper bound on cell $(i, j)$ will be denoted by $b_{ij}$, and the lower bound is zero.

We employ a correspondence between cell-bounded contingency tables and integer flows in a bipartite network. For given values of $r$, $c$ and $b$, consider a network $\mathcal{N}$ defined by $V = [m] \uplus [n]$ and $A = [m] \times [n]$. Then $n = m + n$, $m \le mn$. We will also define $d = m - n + 1 \le (m - 1)(n - 1)$. The supplies $\rho$ are $r_i$ ($i \in [m]$) and $-c_j$ ($j \in [n]$). The capacities are $b(i, j) = b_{ij}$ for $(i, j) \in A$. It is easy to see that there is a bijection between $\Sigma_{r,c,b}$ and the set of integer flows in $\mathcal{N}$.

We are interested in developing an *fpras* for counting cell-bounded contingency tables in for certain classes of inputs. That is, given an error tolerance $\epsilon \in (0, \frac{1}{2})$, the goal is to design an algorithm which runs in time polynomial in $n, \epsilon^{-1}$ and $\max_{ij} \log b_{ij}$ and produces an estimate $|\Sigma'|$ of $|\Sigma_{r,c,b}|$ that satisfies

$$(1 - \epsilon)|\Sigma_{r,c,b}| \ \le\ |\Sigma'| \ \le\ (1 + \epsilon)|\Sigma_{r,c,b}|,$$

with high probability.

The correspondence between flows and cell-bounded tables gives us the following theorem as an immediate consequence of Corollary 3. This can be viewed as a generalisation of a result of Dyer Kannan and Mount [15], with the additional assumption that all cell bounds, as well as row and column sums, are sufficiently large.[1] Note that, since this problem is self-reducible, the existence of an *fpaus* will then follow from the general results of [19].

THEOREM 4. *Let $\Sigma_{r,c,b}$ be a set of cell-bounded contingency tables with row sums $r_i > 0, (i \in [m])$, column sums $c_j > 0, (j \in [n])$, and (tight) cell bounds $b_{ij} > 0$. There is an* fpras *for $|\Sigma_{r,c,b}|$ if $\min_{i,j} b_{ij} \in \Omega((mn)^3/\log(mn))$.*

We note that the result of [20] referred to above would allow us to improve this, to give an fpras under the weaker condition $\min_{i,j} b_{ij} \in \Omega((mn)^{5/2}\sqrt{\log(mn)})$.

In the remainder of the paper we consider a second class of cell-bounded contingency tables where the number of rows $m$ is a *constant*. We show the following theorem, which requires no assumptions about the size of the row and column sums. This theorem also implies the existence of an *fpaus*.

THEOREM 5. *Let $\Sigma_{r,c,b}$ be a set of cell-bounded contingency tables with row sums $r_i > 0, (i \in [m])$, column sums $c_j > 0, (j \in [n])$, and cell bounds $b_{ij} > 0$, and let $m$ be a constant. Then there is an* fpras *for $|\Sigma_{r,c,b}|$.*

The algorithm used to establish Theorem 5 is based on combining the approach of §2 with partial dynamic programming. A critical parameter of the algorithm will be

$$\beta = 6 + 2\log_n(32m^6\epsilon^{-1}).$$

[1]When Corollary 3 is specialised to standard contingency tables, the lower bounds on row and column sums that we require are larger than in [15]. Combining our Theorem 1 with [20], we obtain $\Omega\big((mn)^{5/2}\sqrt{\log(mn)}\big)$, whereas [15] has $\Omega(mn^2)$ for row totals and $\Omega(m^2n)$ for column totals. It is possible that these results could be improved by a more tailored argument or by applying the result of Morris [22], but we will not explore this further in this paper.

The parameters $p, q$ and $r$ will also be important later, with $p = q + \beta/2,\ q \geq \beta,\ r \geq q + \beta$.

We define $r(\ell)$ to be the list of row sums $(r_1, \ldots, r_\ell)$ for any $\ell \leq m$ and $c(k)$ to be the list of column sums $(c_1, \ldots, c_k)$ for any $k \leq n$. We define $r(\overline{\ell})$ to be $(r_{\ell+1}, \ldots, r_m)$ for any $\ell < m$ and $c(\overline{k}) = (c_{k+1}, \ldots, c_n)$ for any $k < n$.

As in §2, we will assume without loss that at the beginning of the dynamic programming phase we have *tight cell bounds*, i.e. for every $i \in [m], j \in [n]$, there is some table $\mathsf{x} \in \Sigma_{r,c,b}$ such that $x_{ij} = b_{ij}$ and some table $\mathsf{y} \in \Sigma_{r,c,b}$ such that $y_{ij} = 0$. Later, it will become necessary to work with networks and cell-bounded tables which do not have tight capacities or bounds. Then we will use $\ell_{ij}$ and $b'_{ij}$ to represent the tight lower and upper bounds for cell $(i, j)$.

The maximum spanning tree $\mathsf{T}$ of the network $\mathcal{N}$ defined in the last section again plays an important role in our algorithm. We will assume that $\mathcal{N}$ is connected, as in §2. In that case, the maximum spanning tree can be constructed in $O(mn \log n)$ time by a standard algorithm [26, §50]. It is straightforward to show that the maximum spanning tree $\mathsf{T}$ is a set of $n + m - 1$ cells. Since each column must contain an element of $\mathsf{T}$, the tree must have the following structure.

(i) A set $K$ of $k = |K| \leq (m - 1)$ columns containing at least two cells per column, giving $(k+m-1) \leq 2(m-1)$ cells in total. We denote this set of cells by $\mathsf{B}$. For each $j \in K$, there is some $i^*$ with $b_{i^*j} = \max_i b_{ij}$ such that $(i^*, j) \in \mathsf{B}$.

(ii) In each of the remaining $(n - k)$ columns $j \in [m] \setminus K$, there is a single cell $(i^*, j) \in \mathsf{T}$ such that $b_{i^*j} = \max_i b_{ij}$, giving $(n - k)$ cells in total. Denote this set of cells by $\mathsf{D}$.

Clearly $\mathsf{T} = \mathsf{B} \uplus \mathsf{D}$. In the special case of *standard* contingency tables [4] $\mathsf{T}$ has a very special structure: $\mathsf{B}$ contains all cells in the column with $\max_j c_j$ and $\mathsf{D}$ contains all remaining cells in the row with $\max_i r_i$. In cell-bounded tables the structure of $\mathsf{T}$ is less predictable but is crucial for our algorithm. We will use $\mathsf{T}$ to partition the columns of the table into *small* and *large columns*, and the rows of the table into *small* and *large rows*. The key to this partitioning is a "jump" property for the cells of $\mathsf{B}$, which results from the following observation.

Lemma 6. *For every $i \in [m]$, there exists a $j' \in [n]$ such that $(i, j') \in \mathsf{B}$ and $b_{ij'} > r_i/mn$.*

Proof. There exists $(i, j^*) \in \mathsf{T}$ such that $b_{ij^*} = \max_j b_{ij} \geq r_i/n$, since $\mathsf{T}$ is a maximum spanning tree. If $(i, j^*) \in \mathsf{B}$, we are finished. Otherwise, suppose $(i, j^*)$ is in $\mathsf{D}$. Column $j^*$ must contain a cell $(i', j^*)$ with $i' \neq i$ such that $b_{i'j^*} > b_{ij^*}/m$. If not, we would have

$$\sum_{i' \neq i} b_{i'j^*} \ \leq\ (m - 1)b_{ij^*}/m \ \leq\ (m - 1)c_{j^*}/m.$$

Then, for every table $\mathsf{x} \in \Sigma_{r,c,b}$, we would require $x_{ij^*} \geq c_{j^*} - \sum_{i' \neq i} b_{i'j^*} \geq c_{j^*}/m$. This contradicts the tightness of the bounds for cell $(i, j^*)$, since it implies a positive lower bound. Hence there must be at least one $i' \neq i$ such that $b_{i'j^*} \geq b_{ij^*}/m$, and our claim holds. Also, because $(i, j^*) \in \mathsf{D}$, we have $(i', j^*) \notin \mathsf{T}$. But now there is a circuit $\Gamma$ in $\mathsf{B} \cup \{(i, j^*), (i', j^*)\}$ with $(i, j^*) \in \Gamma \cap \mathsf{T}$, $(i', j^*) \in \Gamma \setminus \mathsf{T}$. Clearly there is some cell $(i, j') \in \Gamma \cap \mathsf{B}$ and we must have $b_{ij'} \geq b_{i'j^*}$, since $\mathsf{T}$ is a maximum spanning tree. Thus $b_{ij'} \ \geq\ b_{i'j^*} \ >\ b_{ij^*}/m \ \geq\ r_i/mn$. □

We are now ready to prove the "jump" property.

Lemma 7. *Suppose that $n \geq m$ and we are given $r = (r_1, \ldots, r_m), c = (c_1, \ldots, c_n)$, and tight cell bounds $\{b_{ij} : i \in [m], j \in [n]\}$. Then either it is the case that all $b_{ij} < n^{2m\beta}$, or there is a "jump" $\mathcal{J} = [n^q, n^r]$, with $q \geq \beta,\ q + \beta \leq r \leq 2m\beta$, such that $b_{ij} \notin \mathcal{J}$ for all $(i, j) \in \mathsf{B}$.*

Proof. Let $(i_0, j_0)$ be such that $b_{i_0j_0}$ is a largest cell bound. Assume $b_{i_0j_0} > n^{2m\beta}$, from Lemma 6 there exists $(i_0, j_1) \in \mathsf{B}$ with $b_{i_0j_1} \geq n^{2m\beta-2}$. There are at most $2(m-1)$ cells of $\mathsf{B}$. We may assume that some one cell has bound at most $n^{2\beta}$, otherwise we can set $\mathcal{J} = [n^\beta, n^{2\beta}]$. So assume there exists $(i_2, j_2) \in \mathsf{B}$ such that $b_{i_2j_2} \leq n^{2\beta}$. There are at most $2m - 3$ other cells in $\mathsf{B}$. If there is no jump $\mathcal{J}$, then $\max_{(i,j)\in\mathsf{B}} b_{ij} \ \leq\ n^{2\beta} n^{\beta(2m-3)} \ =\ n^{\beta(2m-1)} \ <\ n^{2m\beta-2} \ <\ b_{i_0j_1}$, a contradiction. □

If $b_{ij} < n^{2m\beta}$ for every $i \in [m], j \in [n]$, then we determine $|\Sigma_{r,c,b}|$ exactly by dynamic programming (see §3.1). We now assume the jump $\mathcal{J}$ exists. For any row $i \in [m]$, let $b_{ij'} = \max_{(i,j)\in\mathsf{B}} b_{ij}$. By Lemma 6, if $b_{ij'} < n^q$, then $r_i < mn^{q+1}$. Alternatively, if $b_{ij'} > n^r$, then $r_i > n^r$. Thus all row totals $r_i$ satisfy either $r_i < mn^{q+1}$ or $r_i > n^r$. Therefore the jump in the cells in $\mathsf{B}$ guaranteed by Lemma 7 induces a corresponding, though smaller, jump in the row totals.

Definition 4. *For any cell with $b_{ij} < n^q$, we call $(i, j)$ a* small cell. *The set of all small cells is denoted by $\mathsf{S}$. The remaining cells $\mathsf{A} \setminus \mathsf{S}$ are called* large cells. *We call any row $i$ a* small row *if every cell $(i, j) \in \mathsf{B}$ is small. We assume without loss that $[\sigma]$ is the set of small rows, for some $\sigma \in [m]$. Thus $r_i < mn^{q+1}$ if $i \in [\sigma]$. Rows $[m] \setminus [\sigma]$, called* large rows, *satisfy $r_i > n^r$.*

In §3.2 we show that the flow in small cells and rows can be set arbitrarily, without greatly influencing the number of flows in the residual table, provided the small row totals are satisfied and the column totals are sufficiently large. We show in §3.1 how the number of solutions for the small rows and small cells can be determined by dynamic programming. However, to ensure that we deal with tables in which all column totals are sufficiently large, we first partition the columns of the table into small and large columns.

Definition 5. *If $b_{ij} < n^{2m\beta}$ for all $i \in [m], j \in [n]$, we define the set of* small columns *to be $[n]$. Otherwise $q$ is defined and $p = q + \beta/2$. Then the set of* small columns *is the set of all columns $j$ satisfying $c_j < n^p$. We assume without loss that these are columns $[\nu]$ for some $\nu \in [n]$.*

Let $\mathsf{R} = \{(i, j) \in \mathsf{A} : \sigma < i \leq m,\ \nu < j \leq n\}$. We refer $\mathsf{R}$ as the *residual table.* In §3.1 we will use dynamic programming to decompose the problem into smaller subproblems on $\mathsf{R}$.

First we eliminate the small columns of the table in polynomial time, to express the value of $|\Sigma_{r,c,b}|$ as the weighted sum of a polynomial number of cell-bounded contingency problems, each of size $m$ by $n - \nu$, on the large columns. Let $S_\nu$ be the set of all feasible partial row sums for the small columns. Thus $S_\nu$ is the set of all ordered partitions $t = (t_1, \ldots, t_m)$ of $\sum_{j=1}^{\nu} c_j$ into $m$ parts such that $\Sigma_{t,c(\nu),b(\nu)} \neq \emptyset$. For every $t \in S_\nu$, we will determine

$$w_t =_{\text{def}} |\Sigma_{t,c(\nu),b}|.$$

Next we consider the subproblems on the large columns. Let $s$ denote $(r - t)$ throughout. For each $t \in S_\nu$, we have row sums $s_i$ ($i \in [m]$) for the table on the large columns. We perform dynamic programming on the small rows ($r_i \leq mn^{q+1}$) to count exactly the total number of assignments to all small rows, given that the partial row sum over the small columns is $t$. Let $S'$ be the set of small cells in $\mathsf{R}$, i.e. $S' = \{(i,j) \in \mathsf{R} : b_{ij} \leq n^q\}$. For the remainder of this section we are working with the large columns, and we abbreviate $\Sigma_{s,c(\overline{\nu}),b}$ to $\Sigma_{s,c,b}$.

Definition 6. *Let $\Sigma_{s(\sigma),*,b}$ be the set of tables on the small rows which have row sums $s_i$ for $i \in [\sigma]$, arbitrary column sums, and satisfy the cell bounds for all $i \in [\sigma], j \in [n] \setminus [\nu]$. Let $S'$ denote the set of small cells in $\mathsf{R}$, and let $\Sigma_{S'}$ be the set of assignments to all cells $(i,j) \in S'$ which satisfy the cell bounds.*

The second step in §3.1 will be to calculate, for every $t \in S_\nu$, the term

$$W_t =_{\text{def}} |\Sigma_{s(\sigma),*,b}| \times |\Sigma_{S'}|.$$

Definition 7. *A partial assignment $\mathsf{x}$ to all small cells and all cells in small columns and rows will be called* good *if it satisfies the cell bounds, each small column $j \in [\nu]$ has sum $c_j$, and each small row $i \in [\sigma]$ has sum $r_i$. Let $\mathcal{G}$ be the set of all good assignments.*

For any $\mathsf{x} \in \mathcal{G}$, let $\mathcal{N}^{\mathsf{x}}$ denote the residual network obtained by fixing the values of the small columns, small rows and small cells to their values in $\mathsf{x}$. Let $\mathcal{P}^{\mathsf{x}}$ be the flow polytope for $\mathcal{N}^{\mathsf{x}}$, and let $\mathcal{I}(\mathcal{P}^{\mathsf{x}})$ be the set of lattice points in $\mathcal{P}^{\mathsf{x}}$. Clearly

$$|\Sigma_{r,c,b}| = \sum_{\mathsf{x} \in \mathcal{G}} |\mathcal{I}(\mathcal{P}^{\mathsf{x}})|. \tag{4}$$

We approximate $|\Sigma_{r,c,b}|$ in the following way. We choose any *fixed* $\mathsf{y} \in \mathcal{G}$. The residual network $\mathcal{N}^y$ is not necessarily connected. However, we make two useful observations:

(a) For every good assignment $\mathsf{x} \in \mathcal{G}$, $\mathcal{N}^{\mathsf{x}}$ has the same component structure.

(b) Every component contains at least two rows, so there are at most $m/2$ components.

Facts (a) and (b) follow indirectly from Theorem 8 in §3.1. Fact (a) follows since Theorem 8 implies that $\mathcal{N}^x$ has the component structure of $\mathsf{R} \setminus \mathsf{S}$ for all $\mathsf{x} \in \mathcal{G}$. Fact (b) follows, since Theorem 8 implies $\mathcal{P}^{\mathsf{x}}$ is full-dimensional for every component, but a one-row table has a unique rational flow. For each component $C$ of $\mathcal{N}^y$, we consider the flow polytope $\mathcal{P}^y_C$ of $\mathcal{N}^y_C$. We use volume estimation [13, 21] to estimate $\mathrm{vol}(\mathcal{P}^y_C)$ within relative error $\epsilon/2m$ in time polynomial in $n$, $\epsilon^{-1}$ and $\log(\max_{ij} b_{ij})$. Denote this estimate by $\widehat{\mathrm{vol}}(\mathcal{P}^y_C)$. Next we define

$$\widehat{\mathrm{vol}}(\mathcal{P}^y) =_{\text{def}} \prod_C \widehat{\mathrm{vol}}(\mathcal{P}^y_C).$$

Finally we estimate $|\mathcal{I}(\mathcal{P}^{\mathsf{x}})|$ by $\widehat{\mathrm{vol}}(\mathcal{P}^y)$, for *every* assignment $\mathsf{x} \in \mathcal{G}$. Note $\Sigma_{\mathsf{x}\in\mathcal{G}} 1 = \Sigma_{t \in S_\nu} w_t W_t$. Therefore our estimate $|\Sigma'|$ of $|\Sigma_{r,c,b}|$ is

$$\begin{aligned} |\Sigma'| \;\; =_{\text{def}} \; \textstyle\sum_{\mathsf{x}\in\mathcal{G}} \widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}) &= \widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}) \textstyle\sum_{\mathsf{x}\in\mathcal{G}} 1 \\ &= \widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}) \textstyle\sum_{t\in S_\nu} w_t W_t. \end{aligned} \tag{5}$$

The $w_t$ will be computed (in polynomial time) in the first dynamic programming phase and the $W_t$ in the second dynamic programming phase (see §3.1). The product $\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}})$ is computed by at most $m/2$ calls to a volume estimation algorithm.

To prove that our algorithm is an *fpras*, we must show that, with high probability,

$$(1-\epsilon)|\Sigma_{r,c,b}| \;\leq\; |\Sigma'| \;\leq (1+\epsilon)|\Sigma_{r,c,b}|.$$

In Theorem 9 we prove the following, where $\mathsf{x}, \mathsf{y} \in \mathcal{G}$ and $C$ is any component of $\mathcal{N}^{\mathsf{y}}$,

(i) $(1-\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C) \;\leq\; |\mathcal{I}(\mathcal{P}^{\mathsf{x}}_C)| \;\leq\; (1+\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C)$.

(ii) $(1-\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C) \;\leq\; \mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C) \;\leq\; (1+\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C)$.

The proof of Theorem 9 depends strongly on the jump $\mathcal{J}$ of Lemma 7, and a careful choice of values for $p$, $q$ and $r$. Combining (i) and (ii), we find that for any $\mathsf{x} \in \mathcal{G}$ and component $C$,

$$(1-\epsilon/2m)^2\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C) \;\leq\; |\mathcal{I}(\mathcal{P}^{\mathsf{x}}_C)| \;\leq (1+\epsilon/2m)^2\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C).$$

By construction we know that, with high probability, the estimate $\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}_C)$ lies within $(1 \pm \epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C)$ for every $C$. Therefore, with high probability, for every $\mathsf{x} \in \mathcal{G}$,

$$(1-\epsilon/2m)^3\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}_C) \;\leq\; |\mathcal{I}(\mathcal{P}^{\mathsf{x}}_C)| \;\leq (1+\epsilon/2m)^3\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}_C).$$

For all $\mathsf{x} \in \mathcal{G}$ we have

$$|\mathcal{I}(\mathcal{P}^{\mathsf{x}})| \;=\; \prod_C |\mathcal{I}(\mathcal{P}^{\mathsf{x}}_C)|.$$

There are at most $m/2$ components so, with high probability,

$$\begin{aligned} (1-\epsilon/2m)^{3m/2}\prod_C \widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}_C) \;&\leq\; |\mathcal{I}(\mathcal{P}^{\mathsf{x}})| \\ &\leq\; (1+\epsilon/2m)^{3m/2}\prod_C \widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}_C). \end{aligned}$$

We have $\epsilon \in (0, \frac{1}{2})$, $(1-\epsilon/2m)^{3m/2} \geq 1-\epsilon$ and $(1+\epsilon/2m)^{3m/2} \leq 1+\epsilon$. Hence for all $\mathsf{x} \in \mathcal{G}$,

$$(1-\epsilon)\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}) \;\leq\; |\mathcal{I}(\mathcal{P}^{\mathsf{x}})| \;\leq\; (1+\epsilon)\widehat{\mathrm{vol}}(\mathcal{P}^{\mathsf{y}}).$$

So by (4) and (5), the value $|\Sigma'|$ lies within $(1 \pm \epsilon)|\Sigma_{r,c,b}|$, and our algorithm is indeed an *fpras*.

## 3.1 Dynamic Programming

**Phase 1:** Compute $w_t = |\Sigma_{t,c(\nu),b}|$ for every $t \in S_\nu$. Recall $s = r - t$. There are two cases where we apply dynamic programming:

(1) If there is some $i \in [m], j \in [n]$ such that $b_{ij} \geq n^{2m\beta}$, we apply dynamic programming to the small columns $[\nu]$ to calculate $|\Sigma_{t,c(\nu),b}|$ for every partition $t \in S_\nu$.

(2) If $b_{ij} < n^{2m\beta}$ for all $i \in [m], j \in [n]$, we apply dynamic programming to calculate $|\Sigma_{r,c,b}|$ exactly. We will refer to this case as the $\nu = n$ case.

We consider each column $h$ ($1 \leq h \leq \nu$) in increasing order, and compute $|\Sigma_{t,c(h),b}|$ for each ordered partition $t \in S_h$. This is similar to the dynamic programming phase of the *fpras* of [4] for standard contingency tables with a constant number of rows.

By definition of a small column, we know that every cell bound is at most $c_j < n^p = n^{q+\beta} \leq n^{(2m-1/2)\beta}$ in the $\nu < n$ case, or less than $n^{2m\beta}$ in the $\nu = n$ case. In either case we have $b_{ij} < n^{2m\beta}$ for every cell in a small column.

If $h = 1$, then $|\Sigma_{t,c(1),b}| = 1$ for every partition $t$ of $c_1$ into $m$ parts which satisfies $t_i \leq b_{i1}$ for all $i \in [m]$. Therefore the cardinality of $S_1$ can be bounded by

$$|S_1| \leq \prod_{i=1}^{m-1}(b_{ij}+1) \leq \prod_{i=1}^{m-1} n^{2m\beta} = n^{2m(m-1)\beta},$$

which is polynomial in $n$ and $\epsilon^{-1}$ for $m$ constant. Thus we can list $S_1$ in polynomial time.

If $2 \leq h \leq \nu$, we use the results from the computation on column $(h-1)$ to compute $|\Sigma_{t,c(h),b}|$. Let $\hat{t}$ denote an element of $S_{h-1}$. Then the dynamic programming recurrence is

$$|\Sigma_{t,c(h),b}| = \sum_{\hat{t}\in S_{h-1}:(t-b_h)\leq\hat{t}\leq t} |\Sigma_{\hat{t},c(h-1),b}|, \qquad (6)$$

since there is a unique extension to column $h$ with values $x_{ih} = t_i - \hat{t}_i$ provided these satisfy $0 \leq x_{ih} \leq b_{ih}$. Therefore we can use the $|\Sigma_{\hat{t},c(h-1),b}|$ values constructed for column $(h-1)$ to compute $|\Sigma_{t,c(h),b}|$ for column $h$.

We now bound the running time of the dynamic programming algorithm. First we bound the number of possible $t \in S_h$. We have $b_{ij} < n^{2m\beta}$ for all $i \in [m]$, $j \in [\nu]$. For any $i \in [m-1]$, $\min\{r_i, \sum_{j=1}^{h} b_{ij}\}$ is an upper bound on $t_i$, for any $t \in S_h$. Therefore any $t \in S_h$ must have $t_i < n^{2m\beta+1}$, since $h \in [n]$.

Therefore the cardinality of $S_h$ can be bounded by

$$|S_h| \leq \prod_{i=1}^{m-1}(\sum_{j=1}^{h} b_{ij}+1) \leq \prod_{i=1}^{m-1} n^{2m\beta+1} = n^{(m-1)(2m\beta+1)}.$$

Thus $S_h$ can be listed in $O(n^{(m-1)(2m\beta+1)})$ time, which is polynomial in $n$ and $\epsilon^{-1}$. For each $t \in S_h$, we can calculate $|\Sigma_{t,c(h),b}|$ using equation (6). There are at most $2n^{(m-1)(2m\beta+1)}$ elements of $S_h$, and for each such element we sum over at most $2n^{(m-1)(2m\beta+1)}$ elements of $S_{h-1}$. Therefore the $h^{\text{th}}$ phase of the dynamic programming algorithm takes $O(n^{2(m-1)(2m\beta+1)})$ time. There are at most $n$ phases of dynamic programming. Therefore the running time of the entire algorithm is bounded above by $O(n^{2(m-1)(2m\beta+1)+1})$, which is polynomial in $n$ and $\epsilon^{-1}$.

If $\nu = n$, then we are done. If $\nu < n-1$, we obtain a polynomial-sized set of weights $\{w_t : t \in S_\nu\}$ with $w_t = |\Sigma_{t,c(\nu),b}|$, such that

$$|\Sigma_{r,c,b}| = \sum_{t\in S_\nu} w_t |\Sigma_{s,c(\overline{\nu}),b}|. \qquad (7)$$

**Phase 2:** Compute $W_t = |\Sigma_{s(\sigma),*,b}| \cdot |\Sigma_{S'}|$ for every $t \in S_\nu$. We assume that there is at least one cell with $b_{ij} > n^{2m\beta}$. Then, since the original cell bounds are tight, there is some $j' \neq j$ such that $b_{ij'} \geq n^{2m\beta-1}$. By definition of $p$, both $j$ and $j'$ are large columns. So there are at least two large columns, i.e. $\nu \leq n-2$.

For the remainder of this section, we omit the $\overline{\nu}$ from $c(\overline{\nu})$, $b(\overline{\nu})$, since we are always working with the large columns $[n]\setminus[\nu]$. For each $t \in S_\nu$, we compute the total number of assignments to the small rows and small cells of the table on the large columns with row sums $s = r - t$. We will use dynamic programming to count the following quantity exactly:

$$|\Sigma_{s(\sigma),*,b}| = \prod_{i=1}^{\sigma} |\Sigma^{(i)}_{s_i,*,b}|, \qquad (8)$$

where $\Sigma^{(i)}_{s_i,*,b} =_{\text{def}} \{\,(z_{\nu+1},\ldots,z_n) : \sum_{j=\nu+1}^{n} z_j = s_i, 0 \leq z_j \leq b_{ij}\}$.

For every $i \in [\sigma]$, we compute $|\Sigma^{(i)}_{s_i,*,b}|$ by dynamic programming. For $\nu+1 \leq h \leq n$, and for every $s'_i \leq s_i$, let $\Sigma^{(i,h)}_{s'_i,*,b} =_{\text{def}} \{\,(z_{\nu+1},\ldots,z_h) : \sum_{j=\nu+1}^{h} z_j = s'_i, 0 \leq z_j \leq b_{ij}\}$. Define $|\Sigma^{(i,\nu)}_{s'_i,*,b}| = 1$. Then we use the following recurrence to compute $|\Sigma^{(i)}_{s_i,*,b}| = |\Sigma^{(i,n)}_{s_i,*,b}|$:

$$|\Sigma^{(i,h+1)}_{s'_i,*,b}| = \sum_{x_{h+1}=0}^{\min\{s'_i, b_{i,h+1}\}} |\Sigma^{(i,h)}_{(s'_i-x_{h+1}),*,b}|. \qquad (9)$$

Every row $i \in [\sigma]$ satisfies $r_i < mn^{q+1}$, so $s_i < mn^{q+1}$. Therefore we consider at most $mn^{q+1}$ values for $s'_i$. For each $s'_i$, we consider at most $b_{i,h+1}+1 \leq mn^{q+1}$ values for $x_{h+1}$. Therefore computing $|\Sigma^{(i,h)}_{s'_i,*,b}|$ for a single value of $s'_i$ requires $O(n^{q+1})$ time. For given $h$, we compute $|\Sigma^{(i,h)}_{s'_i,*,b}|$ for all $s'_i \leq s_i$ in $O(n^{2(q+1)})$ time. There are $O(n)$ values of $h$, so we compute $|\Sigma^{(i)}_{s_i,*,b}|$ in $O(n^{2q+3})$ time. We can compute $|\Sigma^{(i)}_{s_i,*,b}|$ for all $i \in [\sigma]$ in $O(n^{2q+3})$ time, since $\sigma = O(1)$. Thus we can compute $|\Sigma_{s(\sigma),*,b}|$ in $O(n^{2q+3})$ time, using (8).

Finally, consider the set of small cells $S' \subseteq \mathsf{R}$. For every $(i,j) \in S'$, we have $b_{ij} < n^q$. Thus there are at most $b_{ij}+1 \leq n^q$ feasible values for each cell $(i,j) \in S'$. To determine $|\Sigma_{S'}|$, we treat each $(i,j) \in S'$ separately. It can be computed in $O(mn)$ time as:

$$|\Sigma_{S'}| = \prod_{(i,j)\in S'} (b_{ij}+1).$$

Hence we can compute $W_t = |\Sigma_{s(\sigma),*,b}| \cdot |\Sigma_{S'}|$ for all $t \in S_\nu$.

## 3.2 Volume Estimation

We state here some facts which were used earlier to show that our algorithm is an *fpras*. Let $\mathcal{N}$ be the network whose flows correspond to elements of $\Sigma_{r,c,b}$. Consider any $\mathsf{x} \in \mathcal{G}$, and let $\mathcal{N}^{\mathsf{x}}$ be its residual network. Any small cell $(i,j) \in S'$ in $\mathsf{R}$ will have been assigned a value by $\mathsf{x}$. We refer to these as *blocked* cells, the remainder being *unblocked*. By the definitions of small rows, small cells and the small columns, it seems likely that the values assigned by $\mathsf{x}$ should not greatly influence the number of flows in $\mathcal{N}^{\mathsf{x}}$. We will prove that this is in fact the case. We first prove the existence of a "central" rational flow in $\mathcal{N}^{\mathsf{x}}$. For any component $C$ of $\mathcal{N}^{\mathsf{x}}$, let $\mathsf{T}_C = \mathsf{B}_C \uplus \mathsf{D}_C$ be a maximum spanning tree on $C$. Let $\mathsf{T}' = \bigcup_C \mathsf{T}_C$, $\mathsf{B}' = \bigcup_C \mathsf{B}_C$ and $\mathsf{D}' = \bigcup_C \mathsf{D}_C$.

THEOREM 8. *Let* $\mathsf{f}$ *be the rational flow on* $\mathcal{N}$ *defined in §2. Suppose* $\mathsf{x} \in \mathcal{G}$ *yields residual network* $\mathcal{N}^{\mathsf{x}}$. *Then there is a rational flow* $\mathsf{g}^{\mathsf{x}}$ *in* $\mathcal{N}^{\mathsf{x}}$ *such that*

$$\begin{aligned} g^{\mathsf{x}}_{ij} &\in f_{ij} \pm (m+1)n^{p+1}, && (i,j) \in \mathsf{B}', \\ g^{\mathsf{x}}_{ij} &\in f_{ij} \pm m^2 n^{q+1}, && (i,j) \in \mathsf{D}', \\ g^{\mathsf{x}}_{ij} &= f_{ij}, && (i,j) \in \mathsf{R}\setminus\mathsf{T}'. \end{aligned}$$

*Also, the tight bounds for any cell* $(i,j)$ *in* $\mathcal{N}^{\mathsf{x}}$ *satisfy* $\ell_{ij} \leq g^{\mathsf{x}}_{ij} - b_{ij}/4mn$ *and* $b'_{ij} \geq g^{\mathsf{x}}_{ij} + b_{ij}/4mn$.

PROOF. Let $\mathsf{T} = \mathsf{B} \uplus \mathsf{D}$ be the original spanning tree for $\mathcal{N}$. Suppose $C$ is a component of $\mathcal{N}^{\mathsf{x}}$. We first show that there is a maximum spanning tree $\mathsf{T}_C = \mathsf{B}_C \uplus \mathsf{D}_C$ in $C$ of unblocked cells, constructed using the original $b_{ij}$, such that

$b_{ij} > n^p/m$ for every $(i,j) \in \mathsf{D}_C$ and $b_{ij} > n^r$ for every $(i,j) \in \mathsf{B}_C$.

Let $I_C$ be the set of rows and $J_C$ the set of columns spanned by $C$, so $\mathsf{B}_C$ is a subtree of $\mathsf{T}_C$ spanning $I_C$. All cells in $\mathsf{D}_C$ satisfy $b_{ij} > n^p/m$, since $c_j > n^p$ for all $j \in J_C$ and the $b_{ij}$ were tight for $\mathcal{N}^{\mathsf{x}}$. Only cells with $b_{ij} < n^p$ have been removed from $\mathcal{N}$, so all cells in the subtree $\mathsf{B}' = C \cap \mathsf{B}$ of $\mathsf{T}$ spanning $I_C$ remain, since $r > p+3$. Therefore every cell $(i,j) \in \mathsf{B}_C$ also satisfies $b_{ij} > n^r$. Otherwise we can obtain a spanning tree of greater weight by removing any $(i,j) \in \mathsf{B}_C$ with $b_{ij} < n^r$ from $\mathsf{T}_C$, and reconnecting the resulting components by the appropriate edge of $\mathsf{B}'$.

Let $\mathsf{f}$ be the rational flow defined in §2, having slack $b_{ij}/2mn$ on every cell $(i,j) \in \mathsf{A}$. We modify $\mathsf{f}$ to obtain a rational flow $\mathsf{g}^{\mathsf{x}}$ in $\mathcal{N}$ so that $g^{\mathsf{x}}_{ij} = x_{ij}$ for all $(i,j) \notin \mathsf{R} \setminus \mathsf{S}$ and every cell $(i,j) \in \mathsf{R} \setminus \mathsf{S}$ has slack at least $b_{ij}/4mn$.

Let $C$ be any component of $\mathcal{N}^{\mathsf{x}}$. The residual row and column sums of the table on $I_C \times J_C$ depend on $\mathsf{x}$. For every $i \in I_C$, the residual row sums $\hat{r}_i(\mathsf{x})$ satisfy

$$\begin{aligned} r_i \;\geq\; \hat{r}_i(\mathsf{x}) &= r_i - \sum_{j=1}^{\nu} x_{ij} - \sum_{j>\nu,(i,j)\in \mathsf{S}} x_{ij} \\ &\geq r_i - \sum_{j=1}^{\nu} b_{ij} - \sum_{j>\nu,(i,j)\in \mathsf{S}} b_{ij} \\ &> r_i - \nu n^p - (n-\nu) n^q \\ &\geq r_i - n^{p+1}. \end{aligned}$$

The residual sums $\hat{c}_j$ for columns $j \in J_C$ satisfy

$$\begin{aligned} c_j \;\geq\; \hat{c}_j(\mathsf{x}) &= c_j - \sum_{i=1}^{\sigma} x_{ij} - \sum_{i>\sigma,(i,j)\in \mathsf{S}} x_{ij} \\ &\geq c_j - \sum_{i=1}^{\sigma} b_{ij} - \sum_{i>\sigma,(i,j)\in \mathsf{S}} b_{ij} \\ &> c_j - \sigma n^q - (m-\sigma) m n^{q+1} \\ &\geq c_j - m^2 n^{q+1}. \end{aligned}$$

Clearly $|\hat{r}_i(\mathsf{x}) - \hat{r}_i(\mathsf{f})| \leq n^{p+1}$ for $i \in [m] \setminus [\sigma]$ and $|\hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f})| \leq m^2 n^{q+1}$ for $j \in [n] \setminus [\nu]$.

Suppose that we modify $\mathsf{f}$ to obtain a new function $\mathsf{f}^{\mathsf{x}}$, as follows:

$$f^{\mathsf{x}}_{ij} = \begin{cases} x_{ij} & \text{for } (i,j) \notin \mathsf{R} \setminus \mathsf{S} \\ f_{ij} & \text{for } (i,j) \in \mathsf{R} \setminus \mathsf{S} \end{cases}$$

The function $\mathsf{f}^{\mathsf{x}}$ is no longer even a rational flow in $\mathcal{N}$. However, we will modify $\mathsf{f}^{\mathsf{x}}$ to produce a rational flow $\mathsf{g}^{\mathsf{x}}$, by changing only the cells in the trees $\mathsf{T}_C$. First we "correct" the column sums for $\mathsf{f}^{\mathsf{x}}$ in $J_C$. Let $j \in J_C$ be such that $\sum_{i=1}^{m} f^{\mathsf{x}}_{ij} \neq c_j$. Let $i^* \in [m]$ be such that $b_{i^*j} = \max_{i \in I_C} b_{ij}$. We may assume that $(i^*, j) \in \mathsf{T}_C$. Then $b_{i^*j} \geq n^p/m$, so

$$\frac{b_{i^*j}}{2mn} \;\geq\; \frac{n^{p-1}}{2m^2} \;=\; \frac{n^{q+\beta/2-1}}{2m^2} \;\geq\; \frac{32m^6n^{q+2}}{2m^2} \;=\; 16m^4n^{q+2}.$$

By definition of $\mathsf{f}$, $\frac{1}{2mn} b_{i^*j} \leq f^{\mathsf{x}}_{i^*j} = f_{i^*j} \;\leq\; (1 - \frac{1}{2mn}) b_{i^*j}$. Thus we can add or subtract $16m^4n^{q+2}$ to $\mathsf{f}^{\mathsf{x}}_{i^*,j}$ and the resulting value satisfies the cell bounds. Since $|\hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f})| \leq m^2 n^{q+1}$, we can add $\hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f})$ to $f^{\mathsf{x}}_{i^*j}$ and maintain the cell bounds for $(i^*, j)$. Since $m^2 n^{q+1} \leq b_{i^*j}/32m^3n^2$, the resulting value in cell $(i,j)$ will still have slack of at least $b_{i^*j}/3mn$. Define ${h^{\mathsf{x}}}_{i^*j} = f_{i^*j} + \hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f})$, $h^{\mathsf{x}}_{ij} = f_{ij}$ $(i \neq i^*)$. Then

$$\begin{aligned} \sum_{i=1}^{m} h^{\mathsf{x}}_{ij} &= \sum_{i=1}^{\sigma} x_{ij} + \sum_{i=\sigma+1}^{m} f_{ij} + \hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f}) \\ &= c_j - \hat{c}_j(\mathsf{x}) + \sum_{i=\sigma+1}^{m} f_{ij} + \hat{c}_j(\mathsf{x}) - \hat{c}_j(\mathsf{f}) \\ &= c_j + \sum_{i=\sigma+1}^{m} f_{ij} - \big(c_j - \sum_{i=1}^{\sigma} f_{ij}\big) \\ &= c_j, \end{aligned}$$

and column total $j$ is satisfied. Note that we corrected column sum $j$ by changing one cell in column $j$. Therefore, we can correct each of the column sums for all $C$ and $j \in J_C$ independently. Letting $\mathsf{T}' = \bigcup_C \mathsf{T}_C$, we obtain a new function $\mathsf{h}^{\mathsf{x}}$ satisfying

$$\begin{aligned} &h^{\mathsf{x}}_{ij} = x_{ij}, && (i,j) \notin \mathsf{R} \setminus \mathsf{S}, \\ &h^{\mathsf{x}}_{ij} = f_{ij}, && (i,j) \in \mathsf{R} \setminus (\mathsf{S} \cup \mathsf{T}'), \\ &h^{\mathsf{x}}_{ij} \in f_{ij} \pm m^2 n^{q+1}, && (i,j) \in \mathsf{T}' \end{aligned}$$

with column sums $c_j$ $(j \in [n])$ and row sums $r_i$ $(i \in [\sigma])$. The row sums for $h^{\mathsf{x}}$ lie in $r_i \pm n^{p+1}$ $(i \in [m] \setminus [\sigma])$, since

$$\begin{aligned} r_i - \nu n^p - (n-\nu) m^2 n^{q+1} &\leq \sum_{j=1}^{n} h^{\mathsf{x}}_{ij} \\ &\leq r_i + \nu n^p + (n-\nu) m^2 n^{q+1}. \end{aligned}$$

Now we correct the row sums for all $C$ and $i \in I_C$. Define $J_C(i) = \{j \in J_C : (i,j) \notin S'\}$ and $I_C(j) = \{i \in I_C : (i,j) \notin S'\}$. Let

$$\mathrm{err}_i(\mathsf{h}^{\mathsf{x}}) \;=\; \sum_{j \in J_C(i)} {h^{\mathsf{x}}}_{ij} - \hat{r}_i(\mathsf{x})$$

be the signed *error* in row $i \in J_C$. Note that

$$\begin{aligned} \sum_{i \in I_C} \mathrm{err}_i(\mathsf{h}^{\mathsf{x}}) &= \sum_{i \in I_C} \sum_{j \in J_C(i)} {h^{\mathsf{x}}}_{ij} - \sum_{i \in I_C} \hat{r}_i(\mathsf{x}) \\ &= \sum_{j \in J_C} \hat{c}_j(\mathsf{x}) - \sum_{i \in I_C} \hat{r}_i(\mathsf{x}) \;=\; 0. \end{aligned}$$

We now correct the row totals by considering pairs of rows $i, i'$ such that $\mathrm{err}_i > 0$ and $\mathrm{err}_{i'} < 0$. Let $\varphi = \min\{\mathrm{err}_i, -\mathrm{err}_{i'}\}$, so $0 < \varphi \leq n^{p+1}$. Let $\mathsf{P}_{i,i'}$ be the unique path in $\mathsf{T}_C$ from $i$ to $i'$. Observe that $\mathsf{P}_{i,i'} \subseteq \mathsf{B}_C$ for every $i, i' \in I_C$, and $b_{kj} \geq n^r$ for every $(k,j) \in \mathsf{B}_C$.

We modify $h^{\mathsf{x}}$ in the cells of $\mathsf{P}_{i,i'}$ by routing flow $\varphi$ from $i$ to $i'$. Note that $\mathsf{P}_{i,i'}$ has odd length, since $\mathcal{N}^{\mathsf{x}}$ is bipartite. List the cells of $\mathsf{P}_{i,i'}$ in order of appearance from $i$ to $i'$, subtract $\varphi$ from all cells in odd positions in this list and add $\varphi$ to cells in even positions. The only change to sums over rows or columns are in rows $i$ and $i'$. Denote the updated solution by $\tilde{\mathsf{h}}^{\mathsf{x}}$. Then

$$\begin{aligned} \mathrm{err}_i(\tilde{\mathsf{h}}^{\mathsf{x}}) &= \mathrm{err}_i(\mathsf{h}^{\mathsf{x}}) - \varphi, \\ \mathrm{err}_{i'}(\tilde{\mathsf{h}}^{\mathsf{x}}) &= \mathrm{err}_{i'}(\mathsf{h}^{\mathsf{x}}) + \varphi, \\ \mathrm{err}_k(\tilde{\mathsf{h}}^{\mathsf{x}}) &= \mathrm{err}_k(\mathsf{h}^{\mathsf{x}}) \quad (k \neq i, i'). \end{aligned}$$

Clearly no error is increased in absolute value. Furthermore, either the new error for row $i$ is zero or the new error for row $i'$ is zero. Thus the routing exactly corrects either row $i$ or row $i'$. We perform this procedure iteratively, at each step choosing a pair of rows $i, i'$ such that the sum in row $i$

has positive error and the sum in row $i'$ is negative. Since we correct at least one row sum at every step, we do this at most $m$ times to obtain a function $\mathsf{g}^{\mathsf{x}}$ with row sums $\hat{r}_i(\mathsf{x})$ and column sums $\hat{c}_j(\mathsf{x})$. The cells which are altered (the $\mathsf{B}_C$ cells) during the row-correcting process still satisfy their cell bounds. We know that the total amount of flow routed from all $i$ to $i'$ pairs is at most $mn^{p+1}$. Therefore no cell changes by more than $mn^{p+1}$ from $\mathsf{h}^{\mathsf{x}}$ to $\mathsf{g}^{\mathsf{x}}$. During the correction of column totals, from $\mathsf{f}^{\mathsf{x}}$ to $\mathsf{h}^{\mathsf{x}}$, an element of $\mathsf{B}_C$ could have been changed once by at most $m^2n^{q+1}$. Thus the total modification to any $\mathsf{B}_C$ cell is at most $mn^{p+1}+m^2n^{q+1}$, which by definition of $\beta, q$, and $p$ is at most $(m+1)n^{p+1}$. But the cells in $\mathsf{B}_C$ have upper bound $b_{ij} > n^r$. Hence, using the definitions of $p, r$, the slack for $\mathsf{g}^{\mathsf{x}}$ in cell $(i,j) \in \mathsf{B}_C$ satisfies

$$\begin{aligned} b_{ij}/2mn \;\; - \;\; & (m+1)n^{p+1} \\ \geq \;\; & b_{ij}/4mn + (n^r/4mn - (m+1)n^{p+1}) \\ \geq \;\; & b_{ij}/4mn, \end{aligned}$$

Thus all cells $(i,j) \in \mathsf{B}_C$ satisfy their cell bounds with slack at least $b_{ij}/4mn$ in $\mathsf{g}^{\mathsf{x}}$. The cells in $\mathsf{D}_C$ are modified only once by at most $m^2n^{q+1}$, in going from from $\mathsf{f}^{\mathsf{x}}$ to $\mathsf{h}^{\mathsf{x}}$. But these cells have upper bound $b_{ij} > n^p/m$. We have already shown that the cells altered in going from $\mathsf{f}^{\mathsf{x}}$ to $\mathsf{h}^{x}$ have slack $b_{ij}/3mn$, therefore these cells will certainly have slack $b_{ij}/4mn$. No other cell is changed. Thus, for every $(i,j) \in \mathsf{T}_C$, the tight bounds have slack at least $b_{ij}/4mn$ in $g^{\mathsf{x}}$. Now we can repeat the argument of §2 to show that, for any cell $(i,j)$, there are rational flows $g', g''$ in $\mathcal{N}^{\mathsf{x}}$ with $g'_{ij} = g^{\mathsf{x}}_{ij} + b_{ij}/4mn$, $g''_{ij} = g^{\mathsf{x}}_{ij} - b_{ij}/4mn$. □

THEOREM 9. *Let $\mathsf{x}, \mathsf{y} \in \mathcal{G}$, let $C$ be any component of* $\mathsf{R}$, *and let $\mathcal{P}^{\mathsf{x}}_C$, $\mathcal{P}^{\mathsf{y}}_C$ be the corresponding flow polytopes. Then*
*(i) $\mathcal{I}(\mathcal{P}^{\mathsf{x}}_C) \in (1 \pm \epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C)$,*
*(ii) $\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C) \in (1 \pm \epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{y}}_C)$.*

PROOF. (i): From Theorem 8, the tight cell bounds satisfy $b'_{ij} - \ell_{ij} \geq b_{ij}/2mn$ for all $(i,j) \in \mathsf{R} \setminus \mathsf{S}$. Now, since

$$\begin{aligned} b'_{ij} - \ell_{ij} \;\geq\; \frac{b_{ij}}{2mn} \;&\geq\; \frac{n^q}{2mn} \\ &\geq\; \frac{n^{6+2\log_n(32m^6\epsilon^{-1})}}{2mn} \\ &>\; \frac{16n^3m^4}{\epsilon} \;=\; (mn)^3\frac{16m}{\epsilon}, \end{aligned}$$

$\mathcal{N}^{\mathsf{x}}_C$ satisfies the conditions of Theorem 2 with $\epsilon = \epsilon/8m$. Therefore, since $1-\theta \leq e^{-\theta}$ and $e^{\theta} < 1+2\theta$ $(0 \leq \theta \leq \frac{1}{2})$,

$$(1-\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C) \;\leq\; \mathcal{I}(\mathcal{P}^{\mathsf{x}}_C) \;\leq\; (1+\epsilon/2m)\mathrm{vol}(\mathcal{P}^{\mathsf{x}}_C).$$

(ii): Observe that $\mathcal{P}^{\mathsf{x}}_C$ is the set of points $z$ satisfying

$$\begin{array}{rclr} \sum_{j\in J_C(i)} z_{ij} &=& \hat{r}_i(\mathsf{x}), & i \in I_C, \\ \sum_{i\in I_C(j)} z_{ij} &=& \hat{c}_j(\mathsf{x}), & j \in J_C, \\ 0 \;\leq\; z_{ij} &\leq& b_{ij}, & (i,j) \in C. \end{array} \tag{10}$$

In Theorem 8 we constructed a rational flow $\mathsf{g}^{\mathsf{x}} \in \mathcal{P}^{\mathsf{x}}_C$ satisfying

$$\tfrac{1}{4mn}b_{ij} \;\leq\; g^{\mathsf{x}}_{ij} \;\leq\; b_{ij}(1-\tfrac{1}{4mn}) \qquad \text{for all } (i,j) \in C,$$

where the $b_{ij}$ are the bounds which were tight for $\mathcal{N}$. They are not necessarily tight for $\mathcal{N}^{\mathsf{x}}_C$, but here we choose to work with the original bounds. Rewriting (10) relative to $\mathsf{g}^{\mathsf{x}}$, by setting $Z_{ij} = z_{ij} - g^{\mathsf{x}}_{ij}$ for all $(i,j) \in C$, gives

$$\begin{array}{rclr} \sum_{j\in J_C(i)} Z_{ij} &=& 0, & i \in I_C, \\ \sum_{i\in I_C(j)} Z_{ij} &=& 0, & j \in J_C, \\ -g^{\mathsf{x}}_{ij} \;\leq\; Z_{ij} &\leq& b_{ij} - g^{\mathsf{x}}_{ij}, & (i,j) \in C. \end{array} \tag{11}$$

For $(i,j) \in \mathsf{D}_C$, let $I'_C(j) = I_C(j) \setminus \{(i,j)\}$. Now, by eliminating $Z_{ij}$, for $(i,j) \in \mathsf{T}_C$, we get the following full-dimensional representation for (11). (See Schrijver [26, §13].)

$$\begin{array}{ll} -g^{\mathsf{x}}_{ij} \leq \sum_{(k,\ell)\in P_{ij}} Z_{k\ell} - \sum_{(k,\ell)\in N_{ij}} Z_{k\ell} & \\ \quad\leq b_{ij} - g^{\mathsf{x}}_{ij}, & (i,j) \in \mathsf{B}_C, \\[2ex] -g^{\mathsf{x}}_{ij} \leq -\sum_{k\in I'_C(j)} Z_{kj} \leq b_{ij} - g^{\mathsf{x}}_{ij}, & \\ & (i,j) \in \mathsf{D}_C, \\[2ex] -g^{\mathsf{x}}_{ij} \leq Z_{ij} \leq b_{ij} - g^{\mathsf{x}}_{ij}, & (i,j) \in C \setminus \mathsf{T}_C. \end{array} \tag{12}$$

where $(k,\ell) \in N_{ij}$ if the directed path in $\mathsf{T}_C$ from row $k$ to column $\ell$ traverses arc $(i,j)$ in the direction $i$ to $j$, and $(k,\ell) \in P_{ij}$ if the directed path in $\mathsf{T}_C$ from $k$ to $\ell$ traverses $(i,j)$ in the direction $j$ to $i$.

Similarly, for any other $\mathsf{y} \in \mathcal{G}$, $\mathcal{P}^{\mathsf{y}}_C$ is the set of points $Z'$ satisfying

$$\begin{array}{ll} -g^{\mathsf{y}}_{ij} \leq \sum_{(k,\ell)\in P_{ij}} Z'_{k\ell} - \sum_{(k,\ell)\in N_{ij}} Z'_{k\ell} & \\ \quad\leq b_{ij} - g^{\mathsf{y}}_{ij}, & (i,j) \in \mathsf{B}_C, \\[2ex] -g^{\mathsf{y}}_{ij} \leq -\sum_{k\in I'_C(j)} Z'_{kj} \leq b_{ij} - g^{\mathsf{y}}_{ij}, & \\ & (i,j) \in \mathsf{D}_C, \\[2ex] -g^{\mathsf{y}}_{ij} \leq Z'_{ij} \leq b_{ij} - g^{\mathsf{y}}_{ij}, & (i,j) \in C \setminus \mathsf{T}_C. \end{array} \tag{13}$$

But our construction of $\mathsf{g}^{\mathsf{x}}$ and $\mathsf{g}^{\mathsf{y}}$ ensures that $\mathsf{g}^{\mathsf{x}}_{ij} = \mathsf{f}_{ij} = \mathsf{g}^{\mathsf{y}}_{ij}$ for every $(i,j) \in C \setminus \mathsf{T}_C$. $Z'_{ij} = Z_{ij}$ for all $(i,j) \in C \setminus \mathsf{T}_C$ and (13) can be written as

$$\begin{array}{ll} -g^{\mathsf{y}}_{ij} \leq \sum_{(k,\ell)\in P_{ij}} Z_{k\ell} - \sum_{(k,\ell)\in N_{ij}} Z_{k\ell} & \\ \quad\leq b_{ij} - g^{\mathsf{y}}_{ij}, & (i,j) \in \mathsf{B}_C, \\[2ex] -g^{\mathsf{y}}_{ij} \leq -\sum_{k\in I'_C(j)} Z_{kj} \leq b_{ij} - g^{\mathsf{y}}_{ij}, & \\ & (i,j) \in \mathsf{D}_C, \\[2ex] -g^{\mathsf{x}}_{ij} \leq Z_{ij} \leq b_{ij} - g^{\mathsf{x}}_{ij}, & (i,j) \in C \setminus \mathsf{T}_C. \end{array} \tag{14}$$

Let $\varepsilon = \epsilon/4m^2n$. We now show $\mathcal{P}^{\mathsf{x}}_C \subseteq (1+\varepsilon)\mathcal{P}^{\mathsf{y}}_C$. It follows from (14) that $(1+\varepsilon)\mathcal{P}^{\mathsf{y}}_C$ is the set of points $Z$ satisfying

$$\begin{array}{ll} -(1+\varepsilon)g^{\mathsf{y}}_{ij} \leq \sum_{(k,\ell)\in P_{ij}} Z_{k\ell} - \sum_{(k,\ell)\in N_{ij}} Z_{k\ell} & \\ \quad\leq (1+\varepsilon)(b_{ij} - g^{\mathsf{y}}_{ij}), & (i,j) \in \mathsf{B}_C, \\[2ex] -(1+\varepsilon)g^{\mathsf{y}}_{ij} \leq -\sum_{k\in I'_C(j)} Z_{kj} & \\ \quad\leq (1+\varepsilon)(b_{ij} - g^{\mathsf{y}}_{ij}), & (i,j) \in \mathsf{D}_C, \\[2ex] -(1+\varepsilon)g^{\mathsf{x}}_{ij} \leq Z_{ij} \leq (1+\varepsilon)(b_{ij} - g^{\mathsf{x}}_{ij}), & \\ & (i,j) \in C \setminus \mathsf{T}_C. \end{array} \tag{15}$$

We must show that every $Z \in \mathcal{P}^{\mathsf{x}}_C$ satisfies (15). Clearly $Z$ satisfies the inequalities for $C \setminus \mathsf{T}_C$. Consider the inequalities for $\mathsf{D}_C$. For $(i,j) \in \mathsf{D}_C$, we have $g^{\mathsf{y}}_{ij}, g^{\mathsf{x}}_{ij} \in f_{ij} \pm m^2n^{q+1}$ by Theorem 8, and therefore $|g^{\mathsf{x}}_{ij} - g^{\mathsf{y}}_{ij}| \leq 2m^2n^{q+1}$. Therefore, we must show that $\varepsilon \min\{b_{ij} - g^{\mathsf{y}}_{ij}, g^{\mathsf{y}}_{ij}\} \geq 2m^2n^{q+1}$. We have $\min\{b_{ij} - g^{\mathsf{y}}_{ij}, g^{\mathsf{y}}_{ij}\} \geq b_{ij}/4mn$ from Theorem 8 and we know that $b_{ij} > n^p/m$ for all $(i,j) \in \mathsf{D}_C$. Thus

$$\begin{aligned} \varepsilon \min\{b_{ij} - g^{\mathtt{y}}_{ij}, g^{\mathtt{y}}_{ij}\} &\geq \frac{\varepsilon n^p}{4m^2 n} = \frac{\epsilon n^{p-2}}{16m^4} \\ &= \frac{\epsilon n^{q+1+\log_n(32m^6\epsilon^{-1})}}{16m^4} \\ &= 2m^2 n^{q+1}, \end{aligned}$$

as required. Finally, consider the cells $(i,j) \in \mathsf{B}_C$. We have $g^{\mathtt{y}}_{ij}, g^{\mathtt{x}}_{ij} \in f_{ij} \pm (m+1)n^{p+1}$ by Theorem 8, so $|g^{\mathtt{y}}_{ij} - g^{\mathtt{x}}_{ij}| \leq 2(m+1)n^{p+1}$. Therefore, we must show that $\varepsilon \min\{b_{ij} - g^{\mathtt{y}}_{ij}, g^{\mathtt{y}}_{ij}\} \geq 2(m+1)n^{p+1}$. We have $\min\{b_{ij} - g^{\mathtt{y}}_{ij}, g^{\mathtt{y}}_{ij}\} \geq b_{ij}/4mn$ from Theorem 8 and we know that $b_{ij} > n^r$ for all $(i,j) \in \mathsf{B}_C$. Thus

$$\begin{aligned} \varepsilon \min\{b_{ij} - g^{\mathtt{y}}_{ij}, g^{\mathtt{y}}_{ij}\} &\geq \frac{\varepsilon n^r}{4mn} = \frac{\epsilon n^{r-2}}{16m^3} \\ &\geq \frac{\epsilon n^{p+1+\log_n(32m^6\epsilon^{-1})}}{16m^3} \\ &= 2m^3 n^{p+1} > 2(m+1)n^{p+1}. \end{aligned}$$

We have now shown that $\mathcal{P}^{\mathtt{x}}_C \subseteq (1+\varepsilon)\mathcal{P}^{\mathtt{y}}_C$. Therefore

$$\begin{aligned} \mathrm{vol}(\mathcal{P}^{\mathtt{x}}_C) &\leq \mathrm{vol}((1+\varepsilon)\mathcal{P}^{\mathtt{y}}_C) \\ &\leq (1+\varepsilon)^{d_C}\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C), \end{aligned}$$

where $d_C = (|I_C|-1)(|J_C|-1) - |S'| \leq mn$ is the dimension of $\mathcal{P}^{\mathtt{x}}_C$ (and $\mathcal{P}^{\mathtt{y}}_C$). Therefore

$$\begin{aligned} \mathrm{vol}(\mathcal{P}^{\mathtt{x}}_C) &\leq (1+\varepsilon)^{mn}\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C) \\ &= \left(1 + \frac{\epsilon}{4m^2 n}\right)^{mn}\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C) \\ &\leq \left(1 + \frac{\epsilon}{2m}\right)\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C), \end{aligned}$$

using $(1+\theta/\kappa)^\kappa \leq 1 + 2\theta$ for $0 \leq \theta \leq \frac{1}{2}$ and $\kappa > 0$.

Switching the roles of $\mathtt{x}$ and $\mathtt{y}$, we also have $\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C) \leq (1+\varepsilon)^{mn}\mathrm{vol}(\mathcal{P}^{\mathtt{x}}_C)$. Then it follows, using $(1+\theta/\kappa)^{-\kappa} \geq 1 - 2\theta$ for $0 \leq \theta \leq \frac{1}{2}$ and $\kappa > 0$, that

$$\begin{aligned} \mathrm{vol}(\mathcal{P}^{\mathtt{x}}_C) &\geq (1+\varepsilon)^{-mn}\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C) \\ &\geq \left(1 - \frac{\epsilon}{2m}\right)\mathrm{vol}(\mathcal{P}^{\mathtt{y}}_C). \end{aligned}$$

□

## 4. REFERENCES

[1] S. Aoki, Exact methods and Markov chain Monte Carlo methods of conditional inference for contingency tables, PhD Thesis, University of Tokyo, 2004.

[2] A. Barvinok, A polynomial-time algorithm for counting integral points in polyhedra when the dimension is fixed, *Mathematics of Operations Research* **19**, pp. 769-779, 1994.

[3] W. Baldoni-Silva, J. De Loera and M. Vergne, Counting integer flows in networks, 2003. Available from `http://arxiv.org/abs/math/0303228`.

[4] M. Cryan and M. Dyer, A polynomial-time algorithm to approximately count contingency tables when the number of rows is constant, in *Journal of Computer and System Sciences*, **67**(2): pp. 291–310, 2003.

[5] M. Cryan, M. Dyer, L. Goldberg, M. Jerrum and R. Martin, Rapidly mixing Markov chains for sampling contingency tables with a constant number of rows, in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 711–720, 2002.

[6] J. De Loera and B. Sturmfels, Algebraic unimodular counting, 2001. Available from `http://arxiv.org/abs/math.CO/0104286`.

[7] P. Diaconis and B. Efron, Testing for independence in a two-way table: new interpretations of the chi-square statistic (with discussion), *Annals of Statistics* **13**, pp. 845–913, 1995.

[8] P. Diaconis and A. Gangolli, Rectangular arrays with fixed margins, in *Discrete probability and algorithms* (D. Aldous, P. Diaconis, J. Spencer and M. Steele, eds.), IMA Volumes on Mathematics and its Applications **72**, Springer-Verlag, New York, pp. 15–41, 1995.

[9] P. Diaconis and L. Saloff-Coste, Random walk on contingency tables with fixed row and column sums, Department of Mathematics, Harvard University, 1995.

[10] P. Diaconis and B. Sturmfels, Algebraic algorithms for sampling from conditional distributions, *Annals of Statistics* **26**, pp. 363–397, 1998.

[11] M. Dyer, Approximate counting by dynamic programming, in *Proc. 35th Annual ACM Symposium on the Theory of Computing*, pp. 693–699, 2003.

[12] M. Dyer and C. Greenhill, Polynomial-time counting and sampling of two-rowed contingency tables. *Theoretical Computer Science* **246**, pp. 265–278, 2000.

[13] M. Dyer, A. Frieze and R. Kannan, A random polynomial time algorithm for approximating the volume of convex bodies, *Journal of the ACM* **38**, pp. 1–17, 1991.

[14] M. Dyer, A. Frieze, R. Kannan, A. Kapoor, L. Perkovic and U. Vazirani, A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem, *Combinatorics, Probability and Computing* **2**, 271-284, 1993.

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

[16] M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, 1991.

[17] R. Holmes and L. Jones, On uniform generation of two-way tables with fixed margins and the conditional volume test of Diaconis and Efron, *Annals of Statistics* **24**, pp. 64–68, 1996.

[18] M. Jerrum, A. Sinclair and E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries, in *Proc. 33rd Annual ACM Symposium on Theory of Computing*, pp. 712–721, 2001.

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

[20] R. Kannan and S. Vempala, Sampling lattice points, in *Proc. of 29th Annual ACM Symposium on Theory of Computing*, pp. 696-700, 1997.

[21] L. Lovász and S. Vempala, Simulated annealing in convex bodies and a $O^*(n^4)$ volume algorithm, in *Proc. 44th Annual IEEE Symposium on Foundations of Computer Science*, pp. 650–659, 2003.

[22] B. Morris, Improved bounds for sampling contingency tables, in *Random Structures & Algorithms*, **21**(2), pp. 135–146, 2002.

[23] J. Mount, *Application of convex sampling to optimization and contingency table generation*, PhD thesis, Carnegie Mellon University, 1995. (Technical Report CMU-CS-95-152, Department of Computer Science.)

[24] J. Mount, Fast unimodular counting, *Combinatorics, Probability and Computing* **9**, pp. 277–285, 2000.

[25] F. Rapollo, Markov bases and structural zeros. Preprint, Department of Mathematics, University of Genova, 2004.

[26] A. Schrijver, *Combinatorial optimization–polyhedra and efficiency*, Springer-Verlag, Berlin, 2003.

[27] L. Valiant, The complexity of computing the permanent, *Theoretical Computer Science* **8**, 189–201, 1979.