

Sampling Binary Contingency Tables with a Greedy Start

Ivona Bezáková*

Nayantara Bhatnagar†

Eric Vigoda†

Abstract

We study the problem of counting and randomly sampling binary contingency tables. For given row and column sums, we are interested in approximately counting (or sampling) $0/1$ $n \times m$ matrices with the specified row/column sums. We present a simulated annealing algorithm with running time $O((nm)^2 D^3 d_{\max} \log^5(n+m))$ for any row/column sums where D is the number of non-zero entries and d_{\max} is the maximum row/column sum. This is the first algorithm to directly solve binary contingency tables for all row/column sums. Previous work reduced the problem to the permanent, or restricted attention to row/column sums that are close to regular. The interesting aspect of our simulated annealing algorithm is that it starts at a non-trivial instance, whose solution relies on the existence of short alternating paths in the graph constructed by a particular Greedy algorithm.

1 Introduction

Counting and randomly sampling binary contingency tables is an important problem in Statistics. Various methods have been proposed in the literature (e.g., see [6, 1, 4, 17] for recent work related to the classical problem of “Darwin’s finches”), but the desired theoretical work on the efficiency of these approaches is lacking.

The problem can be formalized as follows. Given a pair of non-negative integer sequences $r(1), \dots, r(n)$ and $c(1), \dots, c(m)$, our goal is to generate a random $n \times m$ $0/1$ matrix where the i -th row sums to $r(i)$ and the j -th column sums to $c(j)$, for all $1 \leq i \leq n, 1 \leq j \leq m$. An efficient (approximate) sampling scheme then yields an approximation algorithm to count the number of tables with the desired row/column sums. Graph theoretically, we are generating a random bipartite graph with vertex set u_1, \dots, u_n and v_1, \dots, v_m where u_i has degree $r(i)$ and v_j has degree $c(j)$. We will primarily use the graph theoretic view in the presentation and analysis of our algorithm.

Our emphasis is on algorithms that are provably efficient for arbitrary degree sequences. More precisely,

we are seeking an algorithm which approximates the number of tables within a multiplicative factor $1 \pm \varepsilon$ with probability at least $1 - \delta$ and runs in time polynomial in $n, m, 1/\varepsilon$ and $\log(1/\delta)$. Currently the only such method reduces contingency tables to the permanent [11, 10]. The reduction results in a permanent computation of an $\ell \times \ell$ matrix where $\ell = \Omega((n+m)^2)$. The quadratic blowup in the size of the instance is particularly unappealing in light of the large running time for permanent algorithms. The best known algorithm for computing the permanent of an $n \times n$ $0/1$ matrix runs in $O(n^7 \log^4 n)$ time [3].

We present a new algorithm for sampling and counting binary contingency tables with arbitrary degree sequences, by directly exploiting the combinatorial structure of the problem. The resulting algorithm is considerably faster than permanent based algorithms, although our new algorithm is still far from practical.

Before giving a high level description of our results, it is important to highlight several alternative approaches for binary contingency tables. Two popular approaches are a Markov chain, known as the Diaconis chain, which walks on the set of tables with the desired row/column sums [7], and importance sampling, e.g., see [6].

The Diaconis chain is known to converge to a random table for regular degree sequences (i.e., $r(i) = c(j)$ for all i, j) and sequences sufficiently close to regular, see Kannan, Tetali and Vempala [13], and Cooper, Dyer and Greenhill [5]. An alternative (non Markov chain) approach for regular degree sequences with degree $O(n^{1/3})$ was presented by Kim and Vu [14]. No theoretical results are known for either of these approaches for arbitrary degree sequences. The importance sampling approach appears to work well in practice, but there are no theoretical results on its efficiency. We refer the reader to [15] for a discussion of importance sampling, and [6] for recent refinements of the approach.

Our new algorithm is inspired by the permanent algorithm of Jerrum, Sinclair, and Vigoda [11], but requires an interesting algorithmic twist. The new algorithmic idea relies on a combinatorial property of bipartite graphs satisfying a given degree sequence.

The basis of our algorithm is a Markov chain

*Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: ivona@cs.uchicago.edu.

†College of Computing, Georgia Institute of Technology, Atlanta, GA 30332. Email: {nand,vigoda}@cc.gatech.edu. Supported by NSF grant CCR-0455666.

which walks on bipartite graphs with the desired degree sequence and graphs with exactly two deficiencies. We say a graph has a deficiency at vertices u_i and v_j if they have degree $r(i) - 1$ and $c(j) - 1$, respectively, and all other vertices have the desired degree. The number of graphs with the desired degree sequence might be exponentially fewer than the number of graphs with two deficiencies. Thus, we need to weight the random walk defined by the Markov chain so that graphs with the desired degree sequence are likely in the stationary distribution.

Let $w(i, j)$ denote the ratio of the number of graphs with the desired degree sequence versus the number of graphs with deficiencies at u_i and v_j . It turns out that given rough approximations to $w(i, j)$, for all i, j , the Markov chain weighted by these ratios quickly reaches its stationary distribution, and samples from the stationary distribution can then be used to get arbitrarily close estimates of $w(i, j)$. This type of bootstrapping procedure for recalibrating the ratios $w(i, j)$ was central to the algorithm for the permanent.

For the permanent there is an analogous Markov chain on perfect matchings and matchings with at most two unmatched vertices (or holes) where the corresponding ratios, denoted as $\hat{w}(i, j)$, are the number of perfect matchings divided by the number of matchings with holes at u_i, v_j . In the case of the permanent, a bootstrapping algorithm for computing the ratios \hat{w} yields a natural simulated annealing algorithm. Consider an unweighted bipartite graph G that we wish to compute the number of perfect matchings of. In the complete bipartite graph, denoted as G_0 , it is trivial to exactly compute the ratios $\hat{w}(i, j)$ for every i, j . From G_0 , we then slightly decrease the weight of edges not appearing in G , giving a new weighted graph G_1 . Using \hat{w} for G_0 we use the bootstrapping to closely estimate \hat{w} for G_1 . Then we, alternately, decrease (slightly) the weight of non-edges of G creating a new graph G_i , and then use the estimates of \hat{w} for G_{i-1} to bootstrap \hat{w} for G_i . A crucial element of this algorithmic approach is that the quantities $\hat{w}(i, j)$ are trivial to compute in the initial graph, which in this case is the complete bipartite graph.

For contingency tables, what is a starting instance where we can easily estimate the ratios $w(i, j)$'s? Recall that our final goal is to sample subgraphs of the complete bipartite graph with given degree sequence. It is not clear that there is some trivial graph which we can use to start the simulated annealing algorithm. This is the key problem we overcome.

We prove that if we construct a graph G^* with the desired degree sequence using a particular Greedy algorithm, we can estimate the ratios $w(i, j)$ in the

weighted complete bipartite graph where edges of G^* have weight 1 and non-edges have sufficiently small weight. The algorithm to estimate the ratios follows immediately from the following property of G^* . For every pair of vertices u_i, v_j , there is a short alternating path between u_i and v_j , or there is no graph with the degree sequence with deficiencies at u_i, v_j . (An alternating path is a path which alternates between edges and non-edges of G^* .) Moreover, the alternating path is of length at most 5, which implies a trivial algorithm to count the number of minimum length alternating paths. This combinatorial fact is the main result of this paper. It is interesting to note that this combinatorial property fails to hold for many natural variants of Greedy and max-flow algorithms for constructing a graph with a specified degree sequence. To the authors knowledge, this is the first example of a simulated annealing algorithm which starts with a non-trivial instance.

The algorithmic consequence of our work is an $O((nm)^2 D^3 d_{\max} \log^5(n+m))$ time algorithm to approximately count the number of bipartite graphs with the desired degree sequence, where $D = \sum r(i) = \sum c(j)$ is the total degree and $d_{\max} = \max\{\max_i r(i), \max_j c(j)\}$ is the maximum degree. In the worst case this translates to an $O(n^{11} \log^5 n)$ algorithm for an $n \times n$ matrix since $D = O(n^2)$ and $d_{\max} = O(n)$. Moreover, we can count subgraphs of any input graph (see Section 2.5 for a discussion of this extension). The following is a precise statement of our main result.

THEOREM 1.1. *For any bipartite graph $G = (U \cup V, E)$ where $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_m\}$, any degree sequence $r(1), \dots, r(n); c(1), \dots, c(m)$, and any $0 < \varepsilon, \eta < 1$, we can approximate the number of subgraphs of G with the desired degree sequence (i.e., u_i has degree $r(i)$ and v_j has degree $c(j)$, for all i, j) in time $O((nm)^2 D^3 d_{\max} \log^5(nm/\varepsilon) \varepsilon^{-2} \log(1/\eta))$ where $D = \sum_i r(i) = \sum_j c(j)$ is the total degree and $d_{\max} = \max\{\max_i r(i), \max_j c(j)\}$ is the maximum degree. The approximation is guaranteed to be within a multiplicative factor $(1 \pm \varepsilon)$ of the correct answer with probability $\geq 1 - \eta$.*

Since we can consider subgraphs of any bipartite graph, the above formulation generalizes the permanent, and matches the running time of the fastest algorithm for the permanent [3], since for the permanent, we have $D = n, d_{\max} = 1$.

The paper is organized as follows. In Section 2 we give the basic definitions and present a high level description of our simulated annealing algorithm. This section aims to motivate our work on the Greedy algorithm. We prove our main result about short alternating

paths in the graph constructed by a particular variant of Greedy in Section 3. We conclude with a breakup of the running time stated in Theorem 1.1 in Section 4.

2 Preliminaries

2.1 Definitions. We use U and V to denote the partitions of vertices of the bipartite graph on $n + m$ vertices. Thus, the desired degree sequence is denoted by r and c where $r : U \rightarrow \mathbf{N}_0, c : V \rightarrow \mathbf{N}_0$.

For every vertex $v \in V(G)$, let $N(v)$ denote its neighborhood set and let $\overline{N}(v) = V \setminus N(v)$ if $v \in U$, and $\overline{N}(v) = U \setminus N(v)$ if $v \in V$. We will use a, u to denote vertices in U and b, v to denote vertices in V .

DEFINITION 2.1. *We say that a bipartite graph with partitions U, V corresponds to the degree sequences $r : U \rightarrow \mathbf{N}_0, c : V \rightarrow \mathbf{N}_0$ if $\deg(a) = r(a)$ for every $a \in U$ and $\deg(b) = c(b)$ for every $b \in V$. A pair of degree sequences r, c is feasible if there exists a corresponding bipartite graph.*

Let \mathcal{P} be the set of all graphs corresponding to r, c . Recall, our overall aim is to approximate $|\mathcal{P}|$. It is easy to construct a graph with the desired degree sequence, or determine that no such graph exists, using a Greedy algorithm (there are many valid variants) or a max-flow algorithm. We study one such variant of Greedy in Section 3. Hence, we can assume that r, c defines a feasible degree sequence.

In our simulated annealing algorithm, graphs with the desired degree sequence, except at two vertices, called holes (or deficiencies), will play a central role. This is akin to the role of near-perfect matchings in algorithms for the permanent.

DEFINITION 2.2. *Let $u \in U, v \in V$ and let r, c be a pair of degree sequences on U, V . We define degree sequences with holes at u, v as follows:*

$$\hat{r}^{(u)}(a) = \begin{cases} r(a) & \text{if } a \neq u \\ r(a) - 1 & \text{if } a = u \end{cases}$$

$$\hat{c}^{(v)}(b) = \begin{cases} c(b) & \text{if } b \neq v \\ c(b) - 1 & \text{if } b = v \end{cases}$$

We say that u, v is a pair of feasible holes for the degree sequences r, c if the pair of sequences $\hat{r}^{(u)}, \hat{c}^{(v)}$ is feasible.

Let $\mathcal{N}(u, v)$ be the set of all graphs corresponding to $\hat{r}^{(u)}, \hat{c}^{(v)}$ where $u \in U, v \in V$, and let $\mathcal{N} = \cup_{u,v} \mathcal{N}(u, v)$. Let $\Omega = \mathcal{P} \cup \mathcal{N}$.

2.2 High-level Algorithm Description. We now give a rough description of the simulated annealing algorithm. This is not the novel aspect of our paper, the algorithmic approach is very much inspired by algorithms for the permanent. Our emphasis in this section is to motivate our main result about the graph constructed by the Greedy algorithm.

The simulated annealing algorithm will consider a sequence of activities on edges of the complete bipartite graph. There will be a subgraph corresponding to the Greedy algorithm which always has activity 1 on each edge, and the other edges will initially have activities $\lambda \approx 0$, and these edges will slowly increase their activities to $\lambda = 1$. More precisely, let G^* denote the graph with the desired degree sequence constructed by Greedy algorithm which is formally defined in Section 3. (The details of this graph are not relevant at this stage.) For a positive parameter λ , we define the activity of edge $e = (x, y), x \in U, y \in V$, as:

$$\lambda(e) = \begin{cases} 1 & \text{if } e \in E(G^*) \\ \lambda & \text{if } e \notin E(G^*) \end{cases}$$

The activity of a graph $G \in \Omega$ is then defined as:

$$\lambda(G) = \prod_{e \in E(G)} \lambda(e) = \lambda^{|E(G) \setminus E(G^*)|}$$

Finally, the activity of a set of graphs is $\lambda(S) = \sum_{G \in S} \lambda(G)$.

2.3 Bootstrapping. A key quantity are the following *ideal weights*:

$$w_\lambda^*(u, v) = \frac{\lambda(\mathcal{P})}{\lambda(\mathcal{N}(u, v))}$$

It turns out that given close approximations to these weights, there is a Markov chain which can be used to efficiently generate samples from \mathcal{P} weighted by λ . Thus, using these ideal weights for $\lambda = 1$ we can efficiently sample graphs with the desired degree sequence. By a standard reduction [12] approximating $|\mathcal{P}|$ can be reduced to sampling almost uniformly at random from \mathcal{P} .

Given a constant factor approximation to the ideal weights w^* , samples from the Markov chain can be used to boost these weights to an arbitrarily close approximation. This is the bootstrapping procedure. The same approach is used for the approximation of the permanent. Using the bootstrapping procedure to refine rough estimates of the ideal weights we can obtain a simulated annealing algorithm for sampling binary contingency tables. We start with λ_0 close to 0 (specifically with $\lambda_0 = \varepsilon(nm)^{-D}$), where, for a

particular choice of G^* , it turns out to be possible to compute a $(1 \pm \varepsilon)$ approximation of the ideal weights $w_{\lambda_0}^*$ in a straightforward manner. We will then raise λ slightly to a new value λ_1 . The ideal weights $w_{\lambda_0}^*$ for λ_0 will be a rough approximation to the ideal weights for λ_1 . Then, we can use the bootstrapping procedure to boost these to get arbitrarily close estimates of $w_{\lambda_1}^*$. We continue to alternately raise λ slightly, and then bootstrap new estimates of the ideal weights. Eventually, λ becomes 1 and we will have a suitable approximation of the ideal weights for $\lambda = 1$.

Algorithms for the permanent use a similar simulated annealing approach, but instead start at the complete bipartite graph and slowly remove edges not appearing in the input graph. We instead start at a graph which depends on the desired degree sequence. We then slowly add in non-edges until we reach the complete bipartite graph. In some sense we are doing a reverse annealing.

2.4 Estimating Initial Weights. Now we can address how we estimate the ideal weights for λ sufficiently small. Note, $\lambda(\mathcal{P})$ and $\lambda(\mathcal{N}(u, v))$ are polynomials in λ . In particular,

$$\lambda(\mathcal{P}) = \sum_{k=0}^D p_k \lambda^{D-k},$$

where p_k denotes the number of graphs corresponding to r, c which contain exactly k edges of G^* . Similarly,

$$\lambda(\mathcal{N}(u, v)) = \sum_{k=0}^{D-1} p_k^{u,v} \lambda^{D-1-k},$$

where $p_k^{u,v}$ is the number of graphs corresponding to $r^{(u)}, c^{(v)}$ which contain exactly k edges of G^* .

For λ sufficiently small, to approximate these polynomials it suffices to determine the lowest non-zero coefficient. The graph G^* constructed by Greedy has degree sequence r, c , and hence it has exactly one subgraph (G^* itself) that has this degree sequence. Thus, the absolute coefficient of $\lambda(\mathcal{P})$ is 1 and we can approximate $\lambda(\mathcal{P})$ by 1. For $u \in U, v \in V$, if $(u, v) \in E(G^*)$, then the subgraph with edges $E(G^*) \setminus (u, v)$ has holes at u, v , and this is the only subgraph with degree sequence $r^{(u)}, c^{(v)}$. In this case we can also approximate $\lambda(\mathcal{N}(u, v))$ by 1.

If $(u, v) \notin E(G^*)$, then there is no subgraph of G^* with holes at u, v , i.e., degree sequence $r^{(u)}, c^{(v)}$. Note, we can not approximate $\lambda(\mathcal{N}(u, v))$ by 0, since we need an approximation that is close within a multiplicative factor. We instead need to determine the lowest non-zero coefficient of the polynomial. The lowest degree with a non-zero coefficient is dependent on the length

of the shortest alternating path between u and v in G^* . We prove that for our particular choice of G^* , for every u, v there is an alternating path from u to v of length at most 5, or u, v are infeasible holes (in which case we do not need to consider their polynomial). Since these alternating paths are so short, in polynomial time we can simply enumerate all possible such paths, and exactly determine the lowest non-zero coefficient, thereby obtaining a good approximation to $\lambda(\mathcal{N}(u, v))$.

This will result in the following lemma:

LEMMA 2.1. *Let r, c be a feasible degree sequence and let $\varepsilon > 0$ and $\lambda \leq \frac{\varepsilon}{(nm)^D}$. There exists a graph G^* (independent of ε and λ) such that for any pair of feasible holes u, v we can compute a weight $w(u, v)$ satisfying*

$$(1 - \varepsilon)w(u, v) \leq w_{\lambda}^*(u, v) \leq (1 + \varepsilon)w(u, v).$$

in time $O(nmd_{\max}^2)$. Overall, the construction of G^ together with the computation of $w(u, v)$ for all feasible holes u, v takes time $O((nmd_{\max})^2)$.*

2.5 Subgraphs of Arbitrary Input Graph. The above high-level algorithm description applies to the contingency tables problem, where we are generating a random subgraph of the complete bipartite graph $K_{n,m}$ with the desired degree sequence. Our approach extends to subgraphs of any bipartite graph $G = (V, E)$.

The general algorithm proceeds as in Section 2.2. Thus, regardless of G , we construct G^* using the Greedy algorithm and approximate the initial weights. For non-edges of G^* , their activity is slowly raised from $\lambda \approx 0$ to $\lambda = 1$. At this stage all edges have activity $\lambda = 1$, and thus we can generate random subgraphs of $K_{n,m}$ with the desired degree sequence. Then for non-edges of G , i.e., $(u, v) \notin E$, we slowly lower their activity from $\lambda(u, v) = 1$ to $\lambda(u, v) \approx 0$.

Lowering the activities is analogous to raising them, and only requires that the weights w^* at the previous activities can be used to bootstrap the weights w^* at the new activities. The algorithm ends with close approximations to the weights w^* for the graph with activities of $\lambda(u, v) = 1$ for all $(u, v) \in E$ and $\lambda(u, v) \approx 0$ for all $(u, v) \notin E$. Thus, we can generate random subgraphs of G with the desired degree sequence.

2.6 Algorithm Analysis Due to space constraints, we have omitted the precise description and analysis of the Markov chain and simulated annealing algorithm. The analysis is technical and builds on the following recent works. The mixing time of the Markov chain is analyzed using the multicommodity flow approach of Diaconis and Stroock [8], and Sinclair [18]. We

define a canonical flow similar to the construction by Cooper, Dyer, and Greenhill [5]. The analysis of the flows uses combinatorial identities motivated by those recently used in [3] for the permanent. We also use the faster cooling schedule of [3] for the simulated annealing algorithm. The details of the algorithm and its analysis can be found in the full version of this paper [2].

3 Greedy graph

In this section we prove that in the graph constructed by Greedy, a standard greedy algorithm with a specific rule to break ties, for all u, v there is a short alternating path from u to v or there is no graph with holes at u, v . This immediately implies Lemma 2.1.

DEFINITION 3.1. *Let $G = (U, V, E)$ be a bipartite graph with partitions U, V and edge set E , and let $u \in U, v \in V$. We say that there exists an alternating path from u to v of length $2k + 1$, if there exists a sequence of vertices $u = w_0, w_1, \dots, w_{2k}, w_{2k+1} = v$ such that $w_{2i} \in U, w_{2i+1} \in V$ and $(w_{2i}, w_{2i+1}) \in E$ for every $i \in \{0, \dots, k\}$, and $(w_{2i-1}, w_{2i}) \notin E$ for every $i \in \{1, \dots, k\}$.*

The Greedy algorithm depends on an ordering of the vertices. We need an ordering which is consistent with the degree sequence in the following sense.

DEFINITION 3.2. *Fix $c : V \rightarrow \mathbf{N}_0$ and let π be a total ordering on V . We say that π is consistent with c , if for every b_1, b_2 with $c(b_1) > c(b_2)$, vertex b_1 precedes b_2 in π (i.e. $b_1 \prec_\pi b_2$).*

We now define the Greedy algorithm which is the focus of our analysis. It can be viewed as a recursive procedure which matches the highest degree vertex in U , say x , to $r(x)$ highest degree vertices in V . Then the procedure recurses on the residual degree sequence obtained from the original sequence by setting the degree of x to zero and decrementing the degrees of all its neighbors, until all residual degrees equal zero. However, we need to specify how to break ties when two vertices have the same residual degree. This turns out to be the crucial aspect of our algorithm. For this purpose we introduce an additional parameter of the algorithm, a preference relation π which is initially consistent with c . For the recursive call we use a relation $\hat{\pi}$ induced by π on the residual sequence \hat{c} . Here is the formal description of the algorithm:

Procedure GREEDY(r, c, π),

where $r : U \rightarrow \mathbf{N}_0, c : V \rightarrow \mathbf{N}_0$ are degree sequences and π is a total ordering on V consistent with c

- Let $G = (U, V, \emptyset)$ be a bipartite graph with partitions U, V and no edges.

- If $\sum_{a \in U} r(a) \neq \sum_{b \in V} c(b)$, return “Sequences not feasible”.
- If $\sum_{a \in U} r(a) = 0$, return G .
- Let $x \in U$ be a vertex for which $r(x)$ is maximum (if there is more than one, choose arbitrarily).
- Let $Y \subseteq V$ be the first $r(x)$ vertices in the ordering π .
- If Y contains a vertex of degree 0, return “Sequences not feasible”.
- For every $y \in Y$, add the edge (x, y) to G .
- Let $\hat{G} := \text{GREEDY}(\hat{r}, \hat{c}, \hat{\pi})$, where

$$\hat{r}(a) = \begin{cases} r(a) & a \in U \setminus x \\ 0 & a = x \end{cases} \quad \hat{c}(b) = \begin{cases} c(b) - 1 & b \in Y \\ c(b) & b \in V \setminus Y \end{cases}$$

and $\hat{\pi}$ is a total ordering on V defined by: $b_1 \prec_{\hat{\pi}} b_2$ if and only if $\hat{c}(b_1) > \hat{c}(b_2)$ or $\hat{c}(b_1) = \hat{c}(b_2)$ and $b_1 \prec_\pi b_2$.

- Add the edges of \hat{G} to G and return G .
-

It can be shown that the Greedy algorithm does indeed output a graph corresponding to the desired degree sequence iff it is feasible (see the full version [2] for the proof). The correctness of the algorithm appears to be a folklore result. A related, standard algorithmic result is a max-flow characterization for the existence of graphs satisfying a given bipartite degree sequence in [9, 16].

Now we are ready to present our main result. We claim that in the graph constructed by Greedy there is a short (constant-length) alternating path between any two *feasible* holes. One such graph is depicted in Figure 1. It shows a pair of feasible vertices a_5, b_5 and an alternating path $a_5, b_2, a_6, b_6, a_2, b_5$ between them of length 5. Notice that the holes a_7, b_7 are infeasible, thus there is no alternating path between them. In contrast with our main result, it is possible to construct a family of graphs which require alternating paths of linear length for certain pairs of holes. Each graph in this family is an output of a greedy algorithm which breaks ties arbitrarily. The construction can be found in [2].

THEOREM 3.1. *Let r, c be a pair of feasible degree sequences and let π be a total ordering on V consistent with c . Let $G = (U, V, E)$ be the graph constructed by GREEDY(r, c, π). Then for any pair of feasible holes $u \in U, v \in V$ in G there exists an alternating path from u to v of length ≤ 5 .*

Proof. We prove the Theorem by induction on the number of non-zero entries in r . In the base case, there is a single non-zero entry in r . For any pair of feasible holes u, v , the non-zero entry is $r(u)$ and G contains

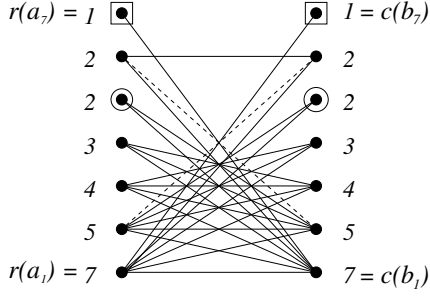


Figure 1: The Greedy graph on the sequence $(1, 2, 2, 3, 4, 5, 7)$, $(1, 2, 2, 3, 4, 5, 7)$.

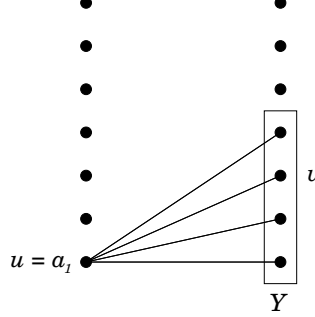


Figure 2: $u = a_1$ and $v \in Y$

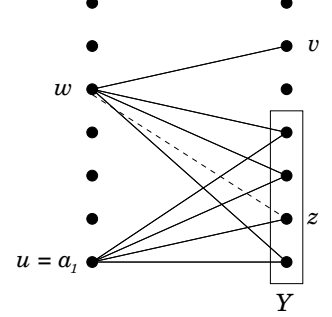


Figure 3: $u = a_1$ and $v \in V \setminus Y$

the edge (u, v) . Thus u, v forms an alternating path of length 1.

For the inductive hypothesis, assume that the Lemma is true for every triple (r', c', π') , where r', c' are feasible degree sequences, r' contains fewer non-zero entries than r , and π' is a total ordering consistent with c' . Let $u \in U, v \in V$ be a pair of feasible holes for r, c . Suppose that $U = \{a_1, \dots, a_n\}$ and the edges adjacent to $a_i \in U$ are added in the i -th iteration (or recursive call) of $\text{GREEDY}(r, c, \pi)$. We say that this is the recursive call when a_i is processed. In the first recursive call $x = a_1$. Recall that Y denotes the set of a_1 's neighbors.

- If $u = a_1$, we construct a short alternating path from u to v as follows (Figures 2,3).
 - If $v \in Y$, then (u, v) is an edge in G and thus u, v forms an alternating path of length 1.
 - If $v \notin Y$, let w be any neighbor of v . Such a neighbor exists since $\deg(v) > 0$, since u, v are feasible holes. Since u is the vertex of the highest degree, $\deg(w) \leq \deg(u)$. Hence there exists a vertex $z \in Y$ which is not a neighbor of w . (If not, then $\deg(w) \geq 1 + |Y| > \deg(u)$, a contradiction.) Then u, z, w, v forms an alternating path of length 3.
- Suppose $u \neq a_1$. Recall that \hat{r}, \hat{c} are the reduced degree sequences corresponding to the graph \hat{G} obtained from G by removing all edges adjacent to a_1 .
 - If u, v are also feasible holes for \hat{r}, \hat{c} , then we may use the inductive hypothesis to conclude that there exists an alternating path from u to v in \hat{G} of length ≤ 5 . Note that the correctness of GREEDY and the assumption that r, c are feasible imply that \hat{r}, \hat{c} are feasible sequences, and $\hat{\pi}$ is consistent with \hat{c} by definition. Hence

we can indeed apply induction. Since G and \hat{G} differ only in edges adjacent to a_1 and the path in \hat{G} does not use a_1 (because $\hat{c}(a_1) = 0$), the path is also an alternating path of length ≤ 5 in G .

- Suppose that u, v are not feasible holes for \hat{r}, \hat{c} . We use the following claim:

CLAIM 1. *If u, v are not feasible holes for \hat{r}, \hat{c} , then $v \in Y$ is of degree $c(v) = 1$ and there exists $v' \in V \setminus Y$ also of degree $c(v') = 1$.*

Before we prove the claim, we check what it implies about the existence of a short alternating path between u and v .

By the claim, $v \in Y$ and there exists another $v' \in V \setminus Y$ with $c(v) = c(v') = 1$. If u has an edge to a vertex $b \in V \setminus Y$, then u, b, a_1, v forms an alternating path of length 3 (see Figure 4). Therefore, we may assume that all of u 's neighbors lie in Y . Let r_j, c_j be the residual degree sequences just before the greedy algorithm for r, c starts adding edges adjacent to $u = a_j$ (i.e., r_j, c_j are GREEDY 's inputs to the recursive call in which u is processed). In other words, r_j, c_j are the parameters of the j -th recursive call originated from $\text{GREEDY}(r, c, \pi)$. Let b be a vertex of the highest remaining degree in $V \setminus Y$ at the start of the j -th recursive call (notice that $V \setminus Y$ is nonempty since $v' \notin Y$). The existence of a short alternating path follows from this claim:

CLAIM 2. *If u, v are feasible, then $c_j(b) = 1$.*

The proof of the claim is included in Section 3.1. By the claim, for feasible u, v there is a vertex $a \in U$ adjacent to b which is processed after u (see Figure 5). This follows from the fact that all of u 's neighbors are in

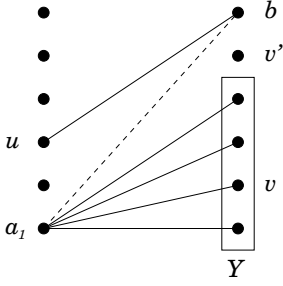


Figure 4: u has a neighbor $b \in V \setminus Y$

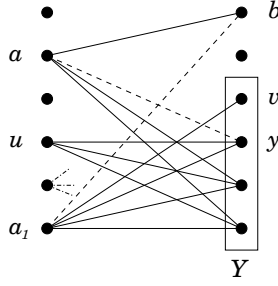


Figure 5: Vertex $b \in V \setminus Y$ of residual degree ≥ 1

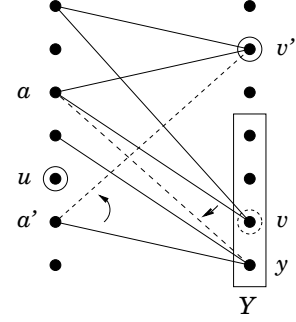


Figure 6: Neighborhoods of v, v' are identical

Y . Hence, $\deg(a) \leq \deg(u)$, and therefore, there exists $y \in Y$ which is a neighbor of u but it is not a neighbor of a . (If not, then $\deg(a) \geq 1 + \deg(u)$, a contradiction.) Then u, y, a, b, a_1, v is an alternating path of length 5. \square

3.1 Proofs of Claims and Main Lemma. To finish the proof of the Theorem 3.1, it remains to prove the two claims. We re-state both claims, together with their assumptions.

CLAIM 1. Recall that \hat{r}, \hat{c} denote the residual sequences after the greedy algorithm matches the first vertex $a_1 \in U$. Assume that u, v are feasible holes for r, c , where $u \neq a_1$. If u, v are not feasible for \hat{r}, \hat{c} , then $v \in Y = N(a_1)$ and there exists a vertex $v' \in V \setminus Y$ such that $c(v) = c(v') = 1$.

Proof. Since u, v are feasible for r, c , there exists a graph with degree sequence $r^{(u)}, c^{(v)}$ (the sequence with holes at u, v). Let $G^{(u,v)}$ be the graph returned by $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi^{(u,v)})$ where $\pi^{(u,v)}$ is the total order obtained from π by repositioning v right after all the vertices of degree $c(v)$ (thus v comes before all vertices of degree $c(v) - 1$). We will compare $G^{(u,v)}$ with G to establish conditions under which u, v are feasible for \hat{r}, \hat{c} .

Notice that if $v \notin Y$ or if $v \in Y$ but $c(v) > c(b)$ for every $b \in V \setminus Y$, then the neighborhoods of a_1 in G and $G^{(u,v)}$ are identical (because the first $r(a_1) = |Y|$ elements of π and $\pi^{(u,v)}$ are the same). Thus, in this case, $\hat{c}^{(v)}$ (the sequence \hat{c} with hole at v) is identical to the sequence $\widehat{c^{(v)}}$, the residual sequence used in the recursive call of $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi^{(u,v)})$. Moreover, since $u \neq a_1$, we have $\widehat{r^{(u)}} = \widehat{r^{(u)}}$. By the correctness of the greedy algorithm, $\widehat{r^{(u)}}, \widehat{c^{(v)}}$ are feasible. Therefore, if a_1 has the same neighbors in G and $G^{(u,v)}$, we can conclude that u, v are feasible holes for \hat{r}, \hat{c} .

We are left with the case when $v \in Y$ and there exists $v' \in V \setminus Y$ of the same degree $c(v') = c(v)$. We will show that if $c(v) > 1$, the holes u, v are feasible for \hat{r}, \hat{c} . This implies the claim.

Suppose $c(v) = c(v') > 1$ for $v \in Y$ and $v' \in V \setminus Y$. Since u, v are feasible for r, c , by symmetry u, v' are also feasible for r, c . However, $v' \notin Y$ and thus, as before, we can conclude that $\hat{r}^{(u)}, \hat{c}^{(v')}$ are feasible and there exists a corresponding graph H . Notice that $\hat{c}^{(v')} = c(v') - 1$ (because $v' \notin Y$ is a hole) and that $\hat{c}^{(v)} = c(v) - 1$ (because $v \in Y$). Since $c(v') = c(v) > 1$, vertices v and v' have the same non-zero degree in H . We will modify H to obtain H' , a graph corresponding to $\hat{r}^{(u)}, \hat{c}^{(v)}$. This will prove the feasibility of u, v for \hat{r}, \hat{c} . To get H' , we need to decrease the degree of v and increase the degree of v' by one while keeping the other degrees intact.

If there is a vertex $a \in U$ which is adjacent to v but not v' in H , we may simply set $H' = H \cup (a, v') \setminus (a, v)$. If there is no such a , then the neighborhood sets of v and v' in H are identical (see Figure 6). If there is a vertex $y \in Y$ for which there exists a neighbor a of v' (and v) in H which is not adjacent to y , then we construct H' as follows. Since $y \in Y$, and $v' \notin Y$ is of the same degree in G as $v \in Y$, by the definition of Y we have $c(y) \geq c(v) = c(v')$. Therefore the degree of y in H is not smaller than the degree of v' in H , i.e. $\hat{c}^{(v')}(y) \geq \hat{c}^{(v)}(v')$. Thus there must exist y 's neighbor a' in H which does not neighbor v' . It suffices to set $H' = H \cup \{(a, y), (a', v')\} \setminus \{(a, v), (a', y)\}$ (see Figure 6).

The last case happens when v and v' share the same set of neighbors in H and every $y \in Y$ is adjacent to every neighbor of v' (see Figure 7). By contradiction we will show that this case never happens. Notice that since $r(a) \leq r(a_1)$ for every $a \in U$ and the degree of a in H remains $r(a)$ except for $u \neq a_1$ which decreases by one, the degree of every $a \in U$ in H is upper bounded by $r(a_1)$, i.e. for all $a \in U$, $\hat{r}^{(u)}(a) \leq |Y|$. Let a

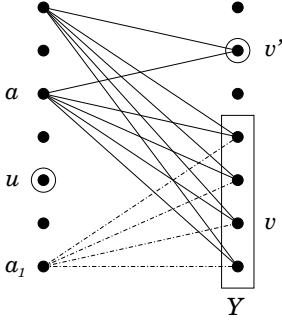


Figure 7: Every $y \in Y$ is adjacent to a

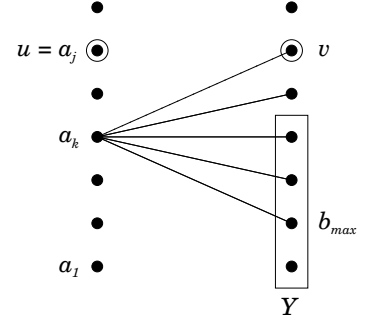
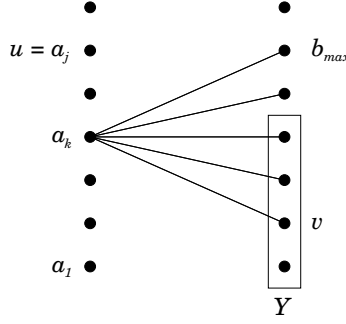


Figure 8: Constructing G and $G^{(u,v)}$ in k -th iteration

be any neighbor of v' (by the assumption $c(v') > 1$, the neighborhood set of v' is non-empty). Then a is adjacent to every vertex in $Y \cup \{v'\}$ and therefore $\hat{r}^{(u)}(a) > |Y|$, a contradiction. \square

CLAIM 2. Let $u \neq a_1$ and $v \in Y$ be such that $c(v) = c(v') = 1$ for some $v' \in V \setminus Y$. Suppose $\text{GREEDY}(r, c, \pi)$ processes vertices from u in order a_1, \dots, a_n . Let r_i, c_i, π_i be the parameters to the i -th recursive call of GREEDY , i.e. the call when a_i is processed. Let $u = a_j$. If $c_j(b) = 0$ for all $b \in V \setminus Y$, then u, v are not feasible for r, c .

Proof. Since $c(v) = 1$, for every $b \in V \setminus Y$ we have $c(b) \leq 1$. Let $b_{\max} \in V \setminus Y$ be the vertex of degree 1 ordered last in π (there is at least one such vertex, since $c(v') = 1$). We create π' by swapping the positions of v and b_{\max} in π . Notice that this ordering is consistent with $c^{(v)}$, the sequence obtained from c by decreasing the degree of v by one. We will compare the execution of $\text{GREEDY}(r, c, \pi)$ and $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$ (see Figure 8). The idea is that both executions will behave similarly, with the roles of v and b_{\max} reversed. Once $\text{GREEDY}(r, c, \pi)$ gets to matching b_{\max} to a vertex a_k from U , $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$ will attempt to match a_k to a vertex in V of residual degree zero and it will fail. Thus, by the correctness of GREEDY , u, v cannot be feasible holes for r, c . We describe the idea in detail below.

Notice that $\text{GREEDY}(r, c, \pi)$ and $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$ process all vertices a_i for $i < j$ in the same order (assume that if the second execution has multiple choices for x , it chooses the same x as the first execution, if possible). This follows from the fact that the only vertex whose degree is changed is vertex a_j and its degree only decreased. Moreover, the order of processing vertices of U is independent of c (or $c^{(v)}$), assuming the executions do not fail.

Let r_i, c_i, π_i and $r_i^{(u)}, c_i^{(v)}, \pi_i'$ be the parameters of the i -th recursive call originated from $\text{GREEDY}(r, c, \pi)$

and $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$, respectively. Let $a_k \in U$ be the vertex matched to b_{\max} by $\text{GREEDY}(r, c, \pi)$. By the assumption of the claim, $c_j(b_{\max}) = 0$, and hence $k < j$, i.e. a_k is processed before $u = a_j$. By induction on i one can verify that for $i < k$ the following hold:

1. $r_i^{(u)}(a) = r_i(a)$ for $a \in U \setminus \{u\}$ and $r_i^{(u)}(u) = r_i(u) - 1$.
2. $c_i^{(v)}(b) = c_i(b)$ for $b \in V \setminus \{v, b_{\max}\}$, $c_i^{(v)}(b_{\max}) = c_i(v)$, $c_i(b_{\max}) = 1$ and $c_i^{(v)}(v) = 0$.
3. Let $\text{supp}(f) = \{x \mid f(x) \neq 0\}$.
 - If $v \in \text{supp}(c_i)$, then $\text{supp}(c_i) = \text{supp}(c_i^{(v)}) \cup \{v\}$. The total order π_i restricted to $\text{supp}(c_i)$ and the total order π_i' restricted to $\text{supp}(c_i^{(v)}) \cup \{v\}$ are identical except that the positions of v, b_{\max} are reversed.
 - If $v \notin \text{supp}(c_i)$, then $\text{supp}(c_i) \setminus \{b_{\max}\} = \text{supp}(c_i^{(v)})$, and the orderings π_i restricted to $\text{supp}(c_i)$ and π_i' restricted to $\text{supp}(c_i^{(v)}) \cup \{v\}$ are identical except that b_{\max} appears in π_i where v appears in π_i' .
4. For every $b \succ_{\pi_i} b_{\max}$, $c_i(b) = 0$. In words, b_{\max} is the last vertex of degree 1 in π_i .

For the induction, the base case $i = 1$ is clear in each case. The case of $i = 2$ also follows in each case, and it can be checked that it holds for the second claim in 3. Now assume the claims are true upto some $2 \leq i \leq k-2$. We show that they hold for $(i+1)$.

1. This part holds since only the degree of $a_i \neq u$ decreases in both sequences by $r_i(a_i) = r_i^{(u)}(a_i)$.
2. Since b_{\max} is matched only in the k -th recursive call, $c_{i+1}(b_{\max}) = c_i(b_{\max}) = 1$. Also, $c_{i+1}^{(v)}(v) = c_i^{(v)}(v) = 0$. For $i = 1$, when v is matched by

the outermost recursive call of $\text{GREEDY}(r, c, \pi)$, b_{\max} is matched by $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$, hence $c_{i+1}^{(v)}(b_{\max}) = c_{i+1}(v)$. We can see that $c_{i+1}^{(v)}(b) = c_{i+1}(b)$ for $b \in V \setminus \{v, b_{\max}\}$ since if the statement is true for i and the orderings are identical on vertices of non-zero residual degree other than v, b_{\max} by 3., then exactly the same set of vertices are used by both in the i -th recursive call.

3. This part is true by the definitions $\pi_{i+1} = \hat{\pi}_i$ and $\pi'_{i+1} = \hat{\pi}'_i$ and the fact that 1, 2, and 3 hold for i .
4. This is clear by the definition of the ordering π_{i+1} and the fact that $c_i(b_{\max}) = 1$.

Therefore a_k is joined to the first $r(a_k)$ elements in π_k by $\text{GREEDY}(r, c, \pi)$, and by 4. the last of them is b_{\max} . However, by 3., the execution of $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$ will attempt to connect a_k to v (or some vertex with remaining degree 0), which is impossible since $c_k^{(v)}(v) = 0$. Thus, $\text{GREEDY}(r^{(u)}, c^{(v)}, \pi')$ fails to construct a corresponding graph, and hence u, v are not feasible holes for r, c . \square

This finishes the proof of Theorem 3.1. As mentioned earlier, Lemma 2.1 is a corollary of Theorem 3.1.

Proof of Lemma 2.1. We will prove that for the greedy graph G^* for any $\varepsilon > 0$ and any $\lambda \leq \frac{\varepsilon}{(nm)^D}$ we can efficiently estimate $w^*(u, v) = \lambda(\mathcal{P})/\lambda(\mathcal{N}(u, v))$ (within a $1 \pm \varepsilon$ factor) for every feasible u, v . We have already observed that $\lambda(\mathcal{P})$ and $\lambda(\mathcal{N}(u, v))$ are polynomials in λ . We will show how to approximate $\lambda(\mathcal{P})$ and $\lambda(\mathcal{N}(u, v))$.

First we observe, that each of $\lambda(\mathcal{P})$ and $\lambda(\mathcal{N}(u, v))$ has a positive small-degree coefficient. In particular, the absolute coefficient of $\lambda(\mathcal{P})$ is 1, since G_0 is the only graph corresponding to r, c sharing exactly D edges with G^* . Moreover, by Theorem 3.1, there exists a graph $G' \in \mathcal{N}(u, v)$ which can be obtained from G^* by swapping the edges of an alternating path of length ≤ 5 . Therefore G' shares at least $D - 3$ edges with G^* and thus the coefficient of x^d for some $d \leq 2$ in $\lambda(\mathcal{N}(u, v))$ is positive. Moreover,

$$|\mathcal{P}| \leq \binom{nm}{D} \leq (nm)^D,$$

where the first inequality follows from the fact that $\binom{nm}{D}$ counts the number of bipartite graphs (with partitions

of sizes n, m) with exactly D edges. Thus,

$$\begin{aligned} 1 \leq \lambda(\mathcal{P}) &= 1 + \sum_{k=0}^{D-1} p_k \lambda^{D-k} \\ &\leq 1 + \lambda \sum_{k=0}^{D-1} p_k \\ &\leq 1 + \lambda(nm)^D \leq 1 + \varepsilon \end{aligned}$$

To approximate $\lambda(\mathcal{N}(u, v))$, we will enumerate all graphs corresponding to $r^{(u)}, c^{(v)}$ which share at least $D - 3$ edges with G^* . This can be done by going through all possible alternating paths from u to v of length ≤ 5 and through all alternating cycles of length 4 (corresponding to the case when the symmetric difference of G^* and the graph with the degree sequence $r^{(u)}, c^{(v)}$ consists of the edge (u, v) and a 4-cycle). This way, for fixed u, v , in time $O(nmd_{\max}^2)$ we can compute

$$x_{u,v} := p_{D-1}^{u,v} + p_{D-2}^{u,v} \lambda + p_{D-3}^{u,v} \lambda^2.$$

We will show that $x_{u,v}$ is a $(1 + \varepsilon)$ -approximation of $\lambda(\mathcal{N}(u, v))$.

$$\begin{aligned} x_{u,v} &\leq \lambda(\mathcal{N}(u, v)) = x_{u,v} + \sum_{k=0}^{D-4} p_k^{u,v} \lambda^{D-1-k} \\ &\leq x_{u,v} + \lambda^3 \sum_{k=0}^{D-4} p_k^{u,v} \\ &\leq x_{u,v} + \varepsilon \lambda^2 \leq (1 + \varepsilon) x_{u,v} \end{aligned}$$

where the last inequality follows from $x_{u,v} \geq \lambda^2$ since there exists $j \in [3]$ for which $p_{D-j}^{u,v} \geq 1$.

Therefore in time $O((nmd_{\max})^2)$ we can compute $x_{u,v}$ for every u, v and $1/x_{u,v}$ is a $(1 + \varepsilon)$ -approximation of $w_\lambda^*(u, v)$. \square

4 Proof of Main Theorem

We now recall the statement of our main theorem, and outline its proof.

THEOREM 1.1. For any bipartite graph $G = (U \cup V, E)$ where $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_m\}$, any degree sequence $r(1), \dots, r(n); c(1), \dots, c(m)$, any $0 < \varepsilon, \eta < 1$, we can approximate the number of subgraphs of G with the desired degree sequence (i.e., u_i has degree $r(i)$ and v_j has degree $c(j)$, for all i, j) in time $O((nm)^2 D^3 d_{\max} \log^5(nm/\varepsilon) \varepsilon^{-2} \log(1/\eta))$ where $D = \sum_i r(i) = \sum_j c(j)$ is the total degree and $d_{\max} = \max\{\max_i r(i), \max_j c(j)\}$ is the maximum degree. And, the approximation is guaranteed to be within a multiplicative factor $(1 \pm \varepsilon)$ of the correct answer with probability $\geq 1 - \eta$.

The Theorem states that we can approximately count the number of bipartite graphs with a given degree sequence which are subgraphs of any given bipartite graph G . In the previous sections we dealt with the case when $G = K_{n,m}$, the complete bipartite graph on $n + m$ vertices. If G is not complete, we can perform the annealing algorithm in two stages. In the first stage, we run the simulated annealing algorithm described previously. Thus, we estimate the ideal weights for $\lambda = 1$ for the complete graph at the end of the first stage. In the second stage, we do the simulated annealing starting with the weights at the end of the first stage (notice that now all edge activities are 1). However, the annealing will *decrease* the activities of edges *not present in G* from 1 to $\lambda \approx 0$ (hence, we may be decreasing the activities of different edges than the ones whose activities were previously increased). The analysis of the annealing algorithm and the mixing time of the Markov chain remain the same. Thus, the two stage process only doubles the running time.

Next, we will give a high-level breakup of the running time for the first stage for which we will assume the results bounding the running time of various stages of the algorithm from [2].

Initially, we spend $O((nmd_{\max})^2)$ time to construct the Greedy graph G^* and to approximate the initial weights, by Lemma 2.1. We need $O(D \log^2(nm))$ intermediate temperatures for the simulated annealing. At each temperature we need to generate $O(nm \log(nm))$ samples from the stationary distribution of the Markov chain in order to do the bootstrapping. Each sample takes $O(D^2 nmd_{\max} \log(nm))$ steps of the Markov chain. With probability $\geq 4/5$ in time $O((nm)^2 D^3 d_{\max} \log^4(nm))$ we can compute correct approximations of the ideal weights w^* for $\lambda = 1$. Therefore, we can generate a random bipartite graph with the desired degree sequence, from a distribution within variation distance $\leq \delta$ of uniform, in time $O((nm)^2 D^3 d_{\max} \log^4(nm/\delta))$. The computation of the initial weights is absorbed by this quantity.

By the reduction from counting to sampling, to get a $1 \pm \varepsilon$ approximation to $|\mathcal{P}|$ takes time $O(D^3 (nm)^2 d_{\max} \log^5(nm/\varepsilon) \varepsilon^{-2})$. Thus, the final running time of the approximate counting algorithm including the weight estimation phase is $O(D^3 (nm)^2 d_{\max} \log^5(nm/\varepsilon) \varepsilon^{-2})$. With probability $\geq 2/3$ the algorithm outputs a $(1 + \varepsilon)$ approximation of the number of bipartite graphs with the desired degree sequence. This can be boosted to probability $\geq 1 - \eta$ by running the algorithm $O(\log \eta^{-1})$ times and outputting the median of the resulting values.

References

- [1] J. Besag and P. Clifford. Generalized Monte-Carlo Significance Tests. *Biometrika*, 76:633-642, 1989.
- [2] I. Bezáková, N. Bhatnagar and E. Vigoda, Sampling Binary contingency Tables with a Greedy Start. *Submitted, available at <http://www.cc.gatech.edu/~vigoda>*.
- [3] I. Bezáková, D. Štefankovič, V. Vazirani, and E. Vigoda, Accelerating Simulated Annealing for Combinatorial Counting. To appear in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.
- [4] G. Cobb and Y. Chen. An Application of Markov Chain Monte-Carlo to Community Ecology. *American Mathematical Monthly*, 110:265-288.
- [5] C. Cooper, M. Dyer, C. Greenhill. On Markov Chains for Random Regular Graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 980-988, 2005.
- [6] Y. Chen, P. Diaconis, S. P. Holmes, and J. S. Liu. Sequential Monte Carlo Methods for Statistical Analysis of Tables. *J. Am. Stat. Assoc.* 100:109-120, 2005.
- [7] P. Diaconis and A. Gangolli, Rectangular Arrays with Fixed Margins. In *Discrete Probability and Algorithms*, eds. D. Aldous et al, New York: Springer-Verlag, 15-41, 1995.
- [8] P. Diaconis, and D. Stroock Geometric bounds for eigenvalues of Markov chains. *Ann. Appl. Probab.*, 1(1):36-61, 1991.
- [9] D. Gale, A theorem on flows in networks, *Pacific J. Math.*, 7:1073-1082, 1957.
- [10] M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *Theor. Comp. Sci.* 73:91-100, 1990.
- [11] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *J. ACM*, 51(4):671-697, 2004.
- [12] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comp. Sci.*, 43:169-188, 1986.
- [13] R. Kannan, P. Tetali, and S. Vempala, Simple Markov-chain algorithms for generating bipartite graphs and tournaments. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 193-200, 1997.
- [14] J. H. Kim and V. H. Vu, Generating random regular graphs. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, 213-222, 2003.
- [15] J. S. Liu, *Monte Carlo Strategies In Scientific Computing*, New York: Springer-Verlag, 2001.
- [16] H. J. Ryser, "Combinatorial Mathematics," Carus Math. Monograph, No. 14, New York, Wiley, 1963.
- [17] J.G. Sanderson. Testing Ecological Patterns. *American Scientist*, 88:332-339, 2000.
- [18] A. Sinclair. Improved Bounds for Mixing Rates of Marked Chains and Multicommodity Flow. *Combinatorics, Probability and Computing*, 1:351-370, 1992.