Algorithmica (2006) 44: 183–193
DOI: 10.1007/s00453-005-1175-9

Algorithmica

Exact Sampling from Perfect Matchings of Dense Regular Bipartite Graphs[1]

Mark Huber[2]

Abstract. We present the first algorithm for generating random variates exactly uniformly from the set of perfect matchings of a bipartite graph with a polynomial expected running time over a nontrivial set of graphs. Previous Markov chain results obtain approximately uniform variates for arbitrary graphs in polynomial time, but their general running time is $\Theta(n^{10}(\ln n)^2)$. Other algorithms (such as Kasteleyn's $O(n^3)$ algorithm for planar graphs) concentrated on restricted versions of the problem. Our algorithm employs acceptance/rejection together with a new upper limit on the permanent of a form similar to Bregman's theorem. For graphs with $2n$ nodes, where the degree of every node is γn for a constant γ, the expected running time is $O(n^{1.5+.5/\gamma})$. Under these conditions, Jerrum and Sinclair showed that a Markov chain of Broder can generate approximately uniform variates in $\Theta(n^{4.5+.5/\gamma} \ln n)$ time, making our algorithm significantly faster on this class of graphs. The problem of counting the number of perfect matchings in these types of graphs is $\sharp P$ complete. In addition, we give a $1+\sigma$ approximation algorithm for finding the permanent of 0–1 matrices with identical row and column sums that runs in $O(n^{1.5+.5/\gamma}(1/\sigma^2)\log(1/\delta))$, where the probability that the output is within $1+\sigma$ of the permanent is at least $1-\delta$.

Key Words. Approximation algorithms, Perfect matchings, Perfect sampling, Permanent, Bregman's theorem, Minc's conjecture.

1. The Problem. Given a bipartite graph (V, E) with n nodes in each of the two partitions (so $|V| = 2n$), a perfect matching is a subset of edges such that each node is incident to exactly one edge. Let Ω denote the set of all such perfect matchings. Generating random variates uniformly from Ω has a variety of applications, such as determining the p values of tests for doubly truncated data [6]. For general graphs, Jerrum et al. [12] gave the first polynomial time algorithm for generating samples that are arbitrarily close to uniform, but their method requires $\Theta(n^{10}(\ln n)^2)$ steps.

Generation algorithms can also be used in constructing algorithms for approximately counting Ω. This problem is equivalent to approximating the permanent of a 0–1 matrix. Computing the permanent of a matrix was one of the first problems shown to be $\sharp P$ complete [20]. Here we consider the restriction where each vertex has at least $n\gamma$ edges and the graph is regular. Under these restrictions, the problem remains $\sharp P$ complete [3].

There exist classes of graphs where the problem is easier. For instance, Kasteleyn showed that for planar graphs $O(n^3)$ operations suffices to count the number of perfect matchings exactly [16].

[1] This research was supported by NSF Postdoc 99-71064.

[2] Department of Mathematics and ISDS, Duke University, Box 90320, Durham, NC 27708-0320, USA. mhuber@math.duke.edu.

Received October 7, 2003; revised January 28, 2005. Communicated by L. Arge.
Online publication September 30, 2005.

Algorithms for approximating the permanent have been constructed using a variety of techniques, including decomposition of the problem [14] and using determinants of related random matrices [1]. The permanent problem is self-reducible (discussed further in Section 2). Using self-reducibility and a technique of Jerrum et al. [13], any polynomial time technique for generating samples that are sufficiently close to the uniform distribution can be used to approximate $|\Omega|$ efficiently, but the number of samples needed can be fairly large. The Jerrum et al. method [12] constructs such an estimate along with a means for generating samples simultaneously—the first sample requires the most time, making the approximation algorithm $\Theta(n^{10}(\ln n)^2)$ as well.

For bipartite graphs with minimum degree $n/2$, the Markov chain of Broder [3] requires $O(n^5 \ln n)$ time to generate approximate samples [5], and will more generally be polynomial when the ratio of the number of almost matchings (matchings with $n-1$ edges) to the number of perfect matchings is polynomial [19].

Given the $\Theta(n^{10}(\ln n)^2)$ technique for generating approximate samples, two questions arise. First, can samples be drawn exactly from the uniform distribution, as opposed to the approximate samples created by Markov chains? Second, can this running time be significantly reduced by restricting the class of graphs we consider?

An exact sampler is preferable to approximate samplers for several reasons. A desirable property of stochastic algorithms is that they be unbiased, that the expected value of the output be equal to the actual output. Finding unbiased algorithms for applications such as estimating p values relies on the output coming exactly from the desired distribution. A more important reason in practice is that algorithms coming from Markov chains are not only $O(T)$, they are $\Theta(T)$ where T is the mixing time of the chain. A $O(T)$ algorithm is always preferable to a $\Theta(T)$ algorithm, because the $O(T)$ algorithm might be faster in practice. The algorithm presented here (as are most exact sampling algorithms) is $O(T)$, and so can be much faster in practice than their $\Theta(T)$ counterparts.

In the remainder of this paper we present an exact sampling technique that is guaranteed to run in polynomial time over a nontrivial set of graphs. Roughly speaking, when the degree of each node is γn, the algorithm takes time $O(n^{1.5+0.5/\gamma})$. The running time of this algorithm is then compared with the running time for Markov chain methods [11] and the estimators that use determinants [9], [15].

As mentioned earlier, self-reducibility can be combined with a (independently derived) method for generating samples to create an approximate counting algorithm. With our method, self-reducibility is used directly to generate the samples and to create an approximate counting algorithm simultaneously. This makes implementation easier, and also makes the running time similar to that needed to draw a sample: the running time is $O(S(1/\sigma^2)\log(1/\delta))$ time, where S is the expected time needed to obtain a single random variate, and the approximation algorithm must come within a factor of $1+\sigma$ of the true answer with probability at least $1-\delta$.

The permanent of a matrix A is defined as

$$per(A) := \sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i\pi(i)}.$$

When the matrix is 0–1 the terms of the permanent are 0 or 1. They are nonzero precisely when $\{\{1, \pi(1)\}, \ldots, \{n, \pi(n)\}\}$ is a perfect matching in the bipartite graph where there

is an edge from the ith node of one partition to the jth node of the other partition if and only if $a_{ij} = 1$. This makes computing the permanent of a 0–1 matrix equivalent to counting the number of perfect matchings.

We will only be able to draw from perfect matchings where the graph is nearly regular. Although our method can be run on instances which are far from being regular and will generate an exact sample when it terminates, it is not guaranteed to terminate in polynomial time unless the graph is nearly regular with approximate degree Δ that is at least γn.

1.1. *Bregman's Theorem.* The heart of our work is a new inequality for permanents that is of the same form as but slightly weaker than Bregman's theorem. Although Bregman's theorem is stronger, it cannot be used algorithmically to generate samples, whereas our new inequality can be used in such a fashion.

First conjectured by Minc [18] and proved by Bregman [2], Bregman's theorem gives the following upper bound on the permanent:

$$M(A) := \prod_{i=1}^{n} (r(i)!)^{1/r(i)} \geq per(A), \tag{1}$$

where $r(i)$ is the sum of the elements of the ith row of A.

Rather than using the Bregman factors $(a!)^{1/a}$, we use the following factors, defined recursively in the following fashion:

$$g(0) := 0, \quad g(1) := e, \quad g(a+1) = g(a) + 1 + \frac{1}{2g(a)} + \frac{0.6}{g(a)^2}, \quad \forall a \geq 1. \tag{2}$$

LEMMA 1. *For all* 0–1 *matrices* A, *with row sums* $\mathbf{r}$,

$$\tilde{M}(A) = \prod_{i=1}^{n} \frac{g(r(i))}{e} \geq per(A). \tag{3}$$

The proof of this lemma is deferred until Section 3 The factors in (3) are fairly close to the factors in Bregman's theorem, but are a slight relaxation:

	a									
	1	2	3	4	5	6	7	8	9	10
Bregman's factor: $(a!)^{1/a}$	1	1.41	1.82	2.21	2.61	2.99	3.38	3.76	4.15	4.53
Relaxed factor: $g(a)/e$	1	1.47	1.89	2.31	2.71	3.11	3.50	3.89	4.27	4.66

Since $g(a+1) \geq g(a) + 1$, it follows that $g(a) \geq a$, which in turn implies for all $a \geq 1$, $g(a+1) - g(a) < 1 + 0.5/a + 0.6/a^2$. Given this fact, it is straightforward to show inductively that for $a \geq 2$,

$$g(a) \leq a + 0.5 \ln a + [g(2) - (2 + 0.5 \ln 2)]. \tag{4}$$

On the other hand, from Stirling's formula

$$\begin{aligned}(a!)^{1/a} &> [\sqrt{2\pi a}(a/e)^a]^{1/a} \\ &= (2\pi a)^{1/(2a)}(a/e) \\ &= (2\pi)^{1/(2a)}e^{\ln a/(2a)}(a/e) \\ &\geq (2\pi)^{1/(2a)}(1+\ln a/(2a))(a/e).\end{aligned}$$

Therefore

$$\lim_{a\to\infty}\frac{g(a)/e}{(a!)^{1/a}} \leq \lim_{a\to\infty}\frac{a+0.5\ln a+g(2)-(2+0.5\ln(2))}{(2\pi)^{1/(2a)}[a+0.5\ln a]}=1, \tag{5}$$

making $g(a)/e$ a reasonable relaxation of the factor $(a!)^{1/a}$.

1.2. *Van der Waerden's Inequality.* To lower bound the number of perfect matchings, we use Van der Waerden's inequality [17], which says that any nonnegative n by n matrix with row and column sums equal to 1 has permanent at least $n!/n^n$. First conjectured by Van der Waerden, it was later proved independently by Falikman [8] and Egorychev [7].

Suppose that our graph is regular, or, equivalently, that all the row and column sums of the matrix A are Δ. Then dividing each row by Δ leaves a new doubly stochastic matrix A' with $per(A') = \Delta^{-n}\, per(A)$. Applying Van der Waerden's inequality to A' gives us $per(A) \geq \Delta^n n!/n^n$. Using Stirling's inequality we have that $n! > n^n e^{-n}\sqrt{2\pi n}$, so our lower bound on the permanent becomes $(\Delta/e)^n\sqrt{2\pi n}$. From (4), the upper bound is $((\Delta+0.5\ln\Delta+g(2)-(2+0.5\ln(2)))/e)^n$. As long as $\Delta \geq \gamma n$, the ratio of the upper to the lower bound will be at most $O(n^{-1/2}n^{1/(2\gamma)})$.

2. The Algorithm. The basic form of the algorithm is not new; it is simply acceptance/rejection. What is new is the form of $\tilde{M}(A)$ that allows us to use the self-reducibility of the permanent problem together with the acceptance/rejection protocol. Effectively, what the algorithm does is choose a number uniformly between 1 and $\tilde{M}(A)$ inclusive. Some of these numbers can be transformed into perfect matching via the algorithm given below. If the number does correspond to a perfect matching, accept, otherwise reject and begin again. We will build up the perfect matching (permutation) corresponding to our random number one entry at a time. That is, we move through the columns one at a time, choosing a different row to go with each column.

Let σ denote our permutation. When choosing row $\sigma(i)$ to go with column i, update the matrix by zeroing out column i and row $\sigma(i)$, except for $A(\sigma(i), i)$ that remains 1. This leaves a smaller version of our original problem. Let $f(A, \sigma(i), i)$ denote this new matrix.

Consider the example of Figure 1. For the first column, we can choose rows 1, 2, or 3 to add to our permutation. Each choice leaves a reduced matrix $f(A, 1, 1)$, $f(A, 2, 1)$, or $f(A, 3, 1)$ from which to choose the remaining permutation. If our random number from 1 to $\tilde{M}(A)$ lies in $\{1, \ldots, \tilde{M}(f(A, 1, 1))\}$, we set $\sigma(1) = 1$. If the number lies in $\{\tilde{M}(f(A, 1, 1)) + 1, \ldots, \tilde{M}(f(A, 1, 1)) + \tilde{M}(f(A, 2, 1))\}$ we set $\sigma(1) = 2$, and so on. If the number does not fall in $\sum_{i=1}^{n} A(i, 1)\tilde{M}(f(A, i, 1))$, then reject. The probability

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad f(A,1,1) = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad f(A,2,1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad f(A,3,1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 1. Example of reduced matrices.

of selecting a particular permutation will be $1/\tilde{M}(A)$, and so conditioned on acceptance the algorithm chooses a permutation with probability $1/per(A)$.

In practice the user will not choose at the beginning of the algorithm a number from 1 to $\tilde{M}(A)$. Instead, the user assigns row $\sigma(j)$ to column j with probability $\tilde{M}(f(A,\sigma(j),j))/\tilde{M}(A)$ at each step. The probability of ending at a given permutation σ is a telescoping product:

$$\frac{\tilde{M}(f(A,\sigma(1),1))}{\tilde{M}(A)} \cdot \frac{\tilde{M}(f(f(A,\sigma(2),2)))}{\tilde{M}(f(A,\sigma(1),1))} \cdots \frac{\tilde{M}(f(f(\cdots f(A,\sigma(n),n))))}{\tilde{M}(f(f(\cdots f(A,\sigma(n-1),n-1))))} = \frac{1}{\tilde{M}(A)}. \tag{6}$$

The pseudocode for this procedure is given in Figure 2.

Important note: to have a valid algorithm we need $\sum_{i:A(i,j)=1} \tilde{M}(f(A,i,j))/\tilde{M}(A) \leq 1$. Otherwise the random choice of $\sigma(j)$ for column j cannot be made with $P(\sigma(j) =$

Generate Random Perfect Matching

Input: Square 0–1 $n \times n$ matrix A
Output: Perfect matching π.

Repeat
  Let ACCEPT $\leftarrow$ TRUE, **Let** $\tilde{A} \leftarrow A$, **Let** $\tilde{\mathbf{r}}$ be the row sums of $\tilde{A}$
  For j from 1 to n
    If $\tilde{r}(i) = 0$ for any i or there exists $i \neq i'$ such that
        $\tilde{A}(i,j) = \tilde{A}(i',j) = r(i) = r(i') = 1$
      Let ACCEPT $\leftarrow$ FALSE
    Else
      Choose R at random from $\{1,\ldots,n+1\}$ using
        $P(R=i) = [\tilde{M}(f(\tilde{A},i,j))/\tilde{M}(\tilde{A})]1(\tilde{A}(i,j)=1)$ for $1 \leq i \leq n$
        $P(R=n+1) = 1 - \sum_i P(R=i)$
      If $1 \leq R \leq n$
        Let $\pi(j) \leftarrow R$
        Let $\tilde{A} \leftarrow f(\tilde{A},R,j)$, **Let** $\tilde{\mathbf{r}}$ be the row sums of $\tilde{A}$
      Else
        Let ACCEPT $\leftarrow$ FALSE
Until ACCEPT = TRUE

Fig. 2. Generate a perfect matching.

$k) = A(k, j)\tilde{M}(f(A, i, j))$. In fact, this is the very reason we use $\tilde{M}(A)$ rather than $M(A)$. Suppose that A is a 5 by 5 matrix whose row sums are all 4 and first column is all 1's. Then $\sum_{i:A(i,j)=1} M(f(A, i, 1)) = 54.5$, but $M(A) = 53.1$. Since $54.5/53.1 > 1$, the algorithm would fail at this point. However, for this example $\sum_{i:A(i,j)=1} \tilde{M}(f(A, i, 1)) = 64.3$, and $\tilde{M}(A) = 65.1$, and so our algorithm proceeds without difficulty. In Section 3 we show in general that $\sum_{i:A(i,j)=1} \tilde{M}(f(A, i, 1)) \leq \tilde{M}(A)$.

2.1. *Estimating the Permanent.* Jerrum et al. showed how to turn a sampling algorithm into an approximate counting algorithm for self-reducible problems [13], but we do not need to employ their method here.

Instead, we simply note that $per(A)/\tilde{M}(A)$ is exactly the probability that the algorithm creates a valid permutation on any given run through the columns. $\tilde{M}(A)$ is of course easy to compute in $O(n \log|\tilde{M}(A)|)$ time; we then just keep track of the number of acceptances over the number of attempts, and multiplying by $\tilde{M}(A)$ gives an approximation to $per(A)$. Standard Chernoff bounds [4] show that after $3\sigma^{-2}\log(1/\delta)$ samples, the estimate will be correct to within a factor of $1 + \sigma$ with probability at least $1 - \delta$.

3. Analyzing the Algorithm. Because the algorithm samples uniformly from the numbers $1, \ldots, \tilde{M}(A)$, when it succeeds the algorithm reaches any particular permutation with probability $1/\tilde{M}(A)$, and conditional on reaching a valid permutation σ (so $A(\sigma(j), j) = 1$ for all j), the probability that the algorithm chooses a particular valid permutation is just $[1/\tilde{M}(A)]/[per(A)/\tilde{M}(A)] = 1/per(A)$ exactly as desired.

Now we show that each step of the algorithm can be successfully performed.

LEMMA 2. *For all* 0–1 *n by n matrices* A, *and* $j \in \{1, \ldots, n\}$,

$$\sum_{i=1}^{n} A(i, j)\tilde{M}(f(A, i, j)) \leq \tilde{M}(A). \tag{7}$$

PROOF. Let $C_j := \{k : A(k, j) = 1\}$. Suppose there exists $i \in C_j$ such that $r(i) = 1$. If $i' \in C_j \backslash \{i\}$ then $\tilde{M}(f(A, i', j))$ contains a factor of $g(0) = 0$. Hence $\sum_{i=1}^{n} A(i, j)\tilde{M}(f(A, i, j)) = \tilde{M}(f(A, i, j))$, and since g is an increasing function and zeroing column j can only reduce the row sums, $\tilde{M}(f(A, i, j)) \leq \tilde{M}(A)$.

Suppose that $r(i) \geq 2$ for all $i \in C_j$. For each such i, $\tilde{M}(f(A, i, j))$ and $\tilde{M}(A)$ each contain many of the same (positive) factors that can be canceled on both sides of the equation. Fix $i \in C_j$. Then the kth factor of $\tilde{M}(A)$ changes from $g(r(i))$ to $g(1)$ and $|C_j| - 1$ row sums of A are reduced by 1 when moving to $\tilde{M}(f(A, i, j))$. Hence

$$\frac{\tilde{M}(f(A, i, j))}{\tilde{M}(A)} = \frac{e}{g(r(i) - 1)} \prod_{k \in C_j} \frac{g(r(k) - 1)}{g(r(k))}. \tag{8}$$

Therefore

$$\frac{\sum_i A(i, j)\tilde{M}(f(A, i, j))}{\tilde{M}(A)} = e\left[\prod_{k \in C_j} \frac{g(r(k) - 1)}{g(r(k))}\right]\left[\sum_{k \in C_j} \frac{1}{g(r(k) - 1)}\right], \tag{9}$$

and we are interested in showing that this expression is bounded above by 1. Rewrite the right-hand side by defining $a_k := g(r(k) - 1)$, for all $k \in C_j$ so that $g(r(k)) = a_k + 1 + 0.5/a_k + 0.6/a_k^2$. Let $\mathbf{a} = (a_1, \ldots, a_n)$ and

$$s(a_k) := 1/a_k, \tag{10}$$

$$p(a_k) := 1/[1 + 1/a_k + 0.5/a_k^2 + 0.6/a_k^3], \tag{11}$$

$$h(\mathbf{a}) := e\left[\prod_{k\in C_j} p(a_k)\right]\left[\sum_{k\in C_j} s(a_k)\right]. \tag{12}$$

Our goal is to show that $h(\mathbf{a})$ is bounded above by 1.

In fact, we will show that $h(\mathbf{a}) \le 1$ for all $\mathbf{a}$ such that $g(1) \le a_k \le g(n-1)$ for all $k \in C_j$. That is we allow $\mathbf{a}$ to vary continuously over the region rather than restricting it to the finite set of values $\{g(1), \ldots, g(n-1)\}$. Since the region is compact and h continuous, h attains its maximum at a particular point $\mathbf{a}^*$.

Suppose that we evaluate h at a vector $\mathbf{a}$ where $a_k < a_\ell$ for some $k \ne \ell$. Let a be any number greater than $[1/a_k + 1/a_\ell]^{-1}$. From $\bar{\mathbf{a}}$ and a, construct a new vector $\bar{\mathbf{a}}$ where $\bar{a}_k = a$ and $\bar{a}_\ell$ is chosen so that

$$s(a_k) + s(a_\ell) = s(a) + s(\bar{a}_\ell). \tag{13}$$

Let $h_{k,\ell}(a) = h(\bar{\mathbf{a}})$. Note $h_{k,\ell}(a)$ can be written as $[s_k(a) + s_\ell(\bar{a}_\ell)]p_k(a)p_\ell(\bar{a}_\ell)T$, where T consists of terms that do not depend on a or $\bar{a}_\ell$. Also, $s_k(a_k) + s_\ell(\bar{a}_\ell)$ is fixed, hence only the $p_k(a)p_\ell(\bar{a}_\ell)$ factors depend on a. Differentiating $s_k(a_k) + s_\ell(\bar{a}_\ell)$ with respect to a shows that $\mathrm{d}\bar{a}_\ell/\mathrm{d}a = -s_k'(a)/s_\ell(\bar{a}_\ell)$. Hence differentiating $h_{k,\ell}(a)$ with respect to a gives

$$h_{k,\ell}'(a) = [h_{k,\ell}(a)/(p(a)p(\bar{a}_\ell))][p'(a)p(\bar{a}_\ell) - p(a)p'(\bar{a}_\ell)s'(a)/s'(\bar{a}_\ell)].$$

Since $s'(x) < 0$ and $p(x) > 0$ over the region of interest,

$$\operatorname{sgn}(h_{k,\ell}'(a)) = \operatorname{sgn}(p'(a)/[p(a)(-s'(a))] - p'(\bar{a}_\ell)/[p(\bar{a}_\ell)(-s'(\bar{a}_\ell))]). \tag{14}$$

Note $h_{k,\ell}'(a)$ has a 0 at $a = \bar{a}_\ell$. Consider the function $t(x) := p'(x)/[p(x)(-s'(x))]$. Using (10) and (11) and simplifying, we find that $t(x) = 1 + [1.3x - 0.6]/[x^3 + x^2 + 0.5x + 0.6]$. Since $x \ge g(1) = e$ the numerator is positive, and the denominator is growing faster than the numerator so $t(x)$ is a decreasing function over our region of interest. This means that $h_{k,\ell}'(a)$ is 0 exactly when $a = \bar{a}_\ell$, is positive when $a < \bar{a}_\ell$, and negative when $a > \bar{a}_\ell$.

Hence $h_{k,\ell}(a)$ has its unique maximum at $a = \bar{a}_\ell$. Since this is true for any k and ℓ with $a_k < a_\ell$, all the components of $\mathbf{a}^*$ are identical.

So instead of maximizing $h(\mathbf{a})$, we can maximize

$$\bar{h}(a) := ep(a)^{c_j}\frac{c_j}{a}. \tag{15}$$

To accomplish this we need two facts about exponentials that are easily verified by taking the appropriate derivatives:

$$(\forall x \ge 0)(1 - x \le \exp\{-(x + 0.5x^2)\}). \tag{16}$$

$$(\forall y \in \mathbf{R})(y\exp\{-y\} \le \exp\{-1\}). \tag{17}$$

Let $\delta(a) := [1/a+0.5/a^2+0.6/a^3]/[1+1/a+0.5/a^2+0.6/a^3]$, so $p(a) = 1-\delta(a)$. Using (16) and (17), we have that

$$\begin{aligned}\bar{h}(a) &\le \frac{e}{a} c_j \exp\{-c_j(\delta(a) + 0.5\delta(a)^2)\} \\ &\le \frac{1}{a(\delta(a) + 0.5\delta(a)^2)} \\ &= 1 - \frac{20a^4 - 5a^3 - 110a^2 - 12a - 72}{200a^6 + 400a^5 + 420a^4 + 435a^2 + 180a^2 + 108a}.\end{aligned}$$

It is easily shown that $20a^4 - 5a^3 - 110a^2 - 12a - 72$ is positive when $a \ge g(1) = e$. Hence $\bar{h}(a) \le 1$, and Lemma 2 is proved. □

3.1. *Running Time.* In this section we derive an upper bound on the expected running time of the algorithm that is polynomial under certain conditions. Even if the graph does not meet these conditions, the algorithm can be used to generate perfect matchings. We only lack a priori bounds on what the running time is.

The time spent inside the repeat loop is easy to compute. Computing the ratios $\tilde{M}(f(\tilde{A}, i, j))/\tilde{M}(\tilde{A})$ takes time $O(n)$. Choosing R takes only $O(n)$ time and $O(\log n)$ expected random bits. Marking the permutation and changing row R and column j to 0 also takes $O(n)$ time. The loop is run n times, so altogether $O(n^2)$ work is needed.

The expected amount of samples taken before acceptance will be $\tilde{M}(A)/per(A)$. We noted in Section 1 that Van der Waerden's inequality [17] together with Stirling's formula may be used to show that $per(A) \ge (\sqrt{2\pi n})(\Delta/e)^n$.

Also noted in Section 1.1: $g(a) \le a + 0.5\ln a + [g(2) - (2 + 0.5\ln 2)]$. With each row sum identically Δ and using $g(2) - (2 + 0.5\ln 2) < 1.65$ gives $\tilde{M}(A) \le ([\Delta + 0.5\ln\Delta + 1.65]/e)^n$. Using $1 + x \le e^x$,

$$\frac{\tilde{M}(A)}{per(A)} \le \frac{1}{\sqrt{2\pi n}} \left(1 + 0.5\frac{\ln\Delta}{\Delta} + \frac{1.65}{\Delta}\right)^n \le \frac{1}{\sqrt{2\pi n}} [5.3\sqrt{\Delta}]^{n/\Delta}.$$

We have proved the following:

THEOREM 1. *The expected running time needed to obtain a sample is*

$$O(n^{1.5}\Delta^{.5n/\Delta}5.3^{n/\Delta}). \tag{18}$$

In particular, if $\Delta = \gamma n$ for some constant γ, then the expected running time is

$$O(n^{1.5+.5/\gamma}).$$

Running time using Markov chains. Jerrum and Sinclair [10] showed that the Markov chain of Broder [3] requires $O(n^2|E|F\ln n)$ time to generate an approximate sample, where $|E|$ is the number of edges in the graph and F is the ratio of almost matchings (with $n-1$ edges) to perfect matchings.

One way to count the number of almost matchings is just to choose which node on the left is unmatched (there are n ways to do this) and then use our algorithm to complete the almost matching as before. This shows that the number of almost matchings is

at most $n\tilde{M}(A)$, hence the ratio of almost matchings to perfect matchings is at most $nM(r)/per(A)$, and the total time needed by the Broder chain will be worse than our algorithm by a factor of $O(n^3 \ln n)$.

Running time using determinant methods. Godsil and Gutman [9] introduced the following method for approximating the permanent of a 0–1 matrix A. Let B be an n by n matrix whose entries are iid draws from $\{-1, 1\}$, and let $C = (c_{ij})$ where $c_{ij} = a_{ij}b_{ij}$. Then let Y be the square of the determinant of C. This is an unbiased estimator for the permanent, that is, $E[Y] = per(A)$. Karmarkar et al. [15] analyzed this method and noted that the running time to obtain an estimate accurate to within $1+\sigma$ with probability at least $1-\delta$ was

$$\Theta\left(\frac{E[Y^2]}{E[Y]^2}\ln\left(\frac{1}{\delta}\right)\frac{1}{\sigma^2}\right).$$

For the case of an n by n matrix of all ones, this ratio was shown to be $E[Y^2]/E[Y]^2 = O(n^2)$.

Karmarkar et al. [15] also introduced another estimator Z where each c_{ij} was uniformly chosen from the three complex roots of unity. The random variable Z is then set to be the magnitude of the complex determinant. Again $E[Z]$ will be the permanent of the original matrix, furthermore, they showed $E[Z^2]/E[Z]^2 = O(n)$ when applied to an n by n matrix of all 1's.

Suppose $\Delta = n\gamma$ as before. Let A_Δ be a Δ by Δ matrix with all entries equal to 1. Then an easy way to create a matrix A with row and column sums Δ is just to make a block matrix where all the diagonal matrices are A_Δ and all off-diagonal matrices are 0:

$$A = \begin{bmatrix} A_\Delta & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & & A_\Delta \end{bmatrix}. \tag{19}$$

The random matrix created in either the Godsil–Gutman or Karmarkar et al. estimator will also have block form and the determinant will be the product of the method applied to the individual blocks. What remains is to bound $E[Z^2]/E[Z]^2$ when the algorithm is applied to a matrix of k by k ones. In [15] it was shown that

$$E[Y^2] = n!\sum_\pi 3^{d(\pi)}, \qquad E[Z^2] = n!\sum_\pi 2^{d(\pi)}, \tag{20}$$

where $d(\pi)$ is the number of cycles in the permutation of length at least 2. Also in [15] it was noted that $\sum_\pi 2^{d(\pi)} \le n!(n+1)$. Next we strengthen the bound on Z since that is the better estimate.

LEMMA 3. *For $d(\pi)$ equal to the number of cycles of length 2 or greater in a permutation π of length n,*

$$n!\frac{n+e-1}{e} \le \sum_\pi 2^{d(\pi)} \le n!\frac{n+e}{e}. \tag{21}$$

PROOF. Consider a uniformly chosen permutation τ on n elements. Let $T(n)$ be the expected value of $2^{d(\tau)}$. Let C be the length of the cycle containing element 1. Then the

number of permutations that have $C = i$ can be found by choosing the $i - 1$ elements of the cycle, writing the cycle in $(i-1)!$ factorial ways, and then deciding the remainder of the permutation in $(n-i)!$ ways. Each permutation has probability $1/n!$ of occurring, so for $i \in \{1, \ldots, n\}$,

$$P(C = i) = \binom{n-1}{i-1}(i-1)!\,(n-i)!\frac{1}{n!} = \frac{1}{n}. \tag{22}$$

Therefore

$$\begin{aligned} T(n) &= E[E[2^{d(\tau)}|C]] \\ &= (1/n)T(n-1) + (1/n)[2T(n-2) + 2T(n-3) + \cdots + 2T(0)], \end{aligned} \tag{23}$$

where $T(0) = T(1) = 1$. Given these initial conditions,

$$(n + e - 1)/e \le T(n) \le (n + e)/e \tag{24}$$

follows from a simple induction. Since $T(n) = (1/n!)\sum_{\pi} 2^{d(\pi)}$, the result follows. □

From this it follows that for the block matrix A defined above:

$$E[Z^2]/E[Z]^2 > (n/e)^{1/\gamma}. \tag{25}$$

Since the determinant takes at least n^2 time to compute, the running time of the determinant algorithm is at least $\Omega(n^{2+1/\gamma}(1/\sigma^2)\ln(1/\delta))$, significantly worse than our algorithm.

3.2. *Estimating the Permanent*. One form of Chernoff's Bound [4] is the following:

THEOREM 2 (Chernoff's Bound). *Let $X_1, \ldots, X_t$ be 0, 1 i.i.d. Bernoulli random variables with parameter p. Then for $\sigma < 1$,*

$$P\left(\left|\sum_i X_i - tp\right| > \sigma tp\right) < 2e^{-\sigma^2 tp/3}.$$

In our algorithm, $p = per(A)/\tilde{M}(r)$. Hence after $O([\tilde{M}(r)/per(A)]\log(1/\delta)/\sigma^2)$ steps, the algorithm will come within $1 + \sigma$ of the true answer with probability at least $1 - \delta$.

THEOREM 3. *The expected running time needed to obtain a $1 + \sigma$ approximation with probability at least $1 - \delta$ is*

$$O(n^{1.5+.5/\gamma}\log(1/\delta)/\sigma^2).$$

4. Conclusions. It can be shown that any bipartite perfect matching problem may be efficiently reduced to a perfect matching problem where the degree of each edge is 3

using ideas of Broder [3]. Further utilizing these techniques, the problem can then be made dense, making the regular problems approximated in this paper $\sharp P$ complete.

It remains to be seen if this approach of sampling from Minc as an upper bound can be proven efficient in situations where the degrees are not regular. The denseness assumption might allow a better lower bound than that given by Van der Waerden.

Our algorithm runs quickly on any matrix where $\tilde{M}(r)/per(A)$ is polynomial. The Jerrum–Sinclair results show that the Markov chain approach runs quickly when the ratio of $(n-1)$-matchings (matchings with only $n-1$ edges) to perfect matchings is polynomial, and it would of course be useful to be able to make precise the connection between these two ratios.

References

[1] A. Barvinok. Polynomial time algorithms to approximate permanents and mixed discriminants within a simply exponential factor. *Random Structures Algorithms*, 14:29–61, 1999.

[2] L. M. Bregman. Some properties of nonnegative matrices and their permanents. *Soviet. Math. Dokl.*, 14(4):945–949, 1973.

[3] A. Z. Broder. How hard is it to marry at random? (On the approximation of the permanent). In *Proc. 18th ACM Sympos. on Theory of Computing*, pages 50–58, 1986.

[4] H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations. *Ann. of Math. Stat.*, 23:493–509, 1952.

[5] P. Diaconis and D. Stroock. Geometric bounds for eigenvalues of Markov chains. *Ann. Appl. Probab.*, 1:36–61, 1991.

[6] B. Efron and V. Petrosian. Nonparametric methods for doubly truncated data. *J. Amer. Statist. Assoc.*, 94(447):824–834, 1999.

[7] G. P. Egorychev. The solution of van der Waerden's problem for pemanents. *Adv. Math.*, 42:299–305, 1981.

[8] D. I. Falikman. Proof of the van der Waerden's conjecture on the permanent of a doubly stochastic matrix. *Mat. Zametki*, 29(6):931–938, 1981.

[9] C. D. Godsil and L. Gutman. On the matching polynomial of a graph. *Colloq. Math. Soc. János Bolyai*, 25:241–249, 1981.

[10] M. Jerrum and A. Sinclair. Approximating the permanent. *J. Comput.*, 18:1149–1178, 1989.

[11] 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 ACM Sympos. on Theory of Computing*, pages 712–721, 2001.

[12] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[13] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoret. Comput. Sci.*, 43:169–188, 1986.

[14] M. Jerrum and U. V. Vazirani. A mildly exponential approximation algorithm for the permanent. *Algorithmica*, 16(4/5):392–401, 1996.

[15] N. Karmarkar, R. Karp, R. Lipton, L. Lovász, and M. Luby. A Monte Carlo algorithm for estimating the permanent. *SIAM J. Comput.*, 22(2):284–293, 1993.

[16] P. W. Kasteleyn. The statistics of dimers on a lattice, I, the number of dimer arrangements on a quadratic lattice. *Physica*, 27:1664–1672, 1961.

[17] J. H. Van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, Cambridge, 1992.

[18] H. Minc. Upper bounds for permanents of (0, 1)-matrices. *Bull. Amer. Math. Soc.*, 69:789–791, 1963.

[19] A. Sinclair. Improved bounds for mixing rates of Markov chains and multicommodity flow. *Combin. Probab. Comput.*, 1:351–370, 1992.

[20] L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8:189–201, 1979.