Biometrika (2009), **96**, 2, *pp*. 1–10
© 2008 Biometrika Trust
Printed in Great Britain
Advance Access publication on 13 February 2008

Approximating the α-permanent

BY S. C. KOU
Department of Statistics, Harvard University, Cambridge, Massachusetts 02138, USA
kou@stat.harvard.edu

AND P. MCCULLAGH
Department of Statistics, University of Chicago, 5734 University Ave., Chicago 60637, USA
pmcc@galton.uchicago.edu

SUMMARY

The standard matrix permanent is the solution to a number of combinatorial and graph-theoretic problems, and the α-weighted permanent is the density function for a class of Cox processes called boson processes. Exact computation of the ordinary permanent is known to be #P-complete, and the same appears to be the case for the α-permanent for most values of α. At present, the lack of a satisfactory algorithm for approximating the α-permanent is a formidable obstacle to the use of boson processes in applied work. This article proposes an importance-sampling estimator using non-uniform random permutations generated in cycle format. Empirical investigation reveals that the estimator works well for the sorts of matrices that arise in point-process applications, involving up to a few hundred points. We conclude with a numerical illustration of the Bayes estimate of the intensity function of a boson point process, which is a ratio of α-permanents.

Some key words: Boson point process; Conditional intensity; Density estimation; Sequential importance sampling.

1. WEIGHTED PERMANENT

The α-weighted permanent of a square matrix A of order n, defined as

$$\mathrm{per}_\alpha(A) = \sum_{\sigma\in\Pi_n} \alpha^{\mathrm{cyc}(\sigma)} \prod_i A_{i,\sigma(i)},$$

is a sum over all permutations of $\{1,\ldots,n\}$, where $\mathrm{cyc}(\sigma)$ is the number of cycles in σ. It is evident that $\mathrm{per}_\alpha(A)$ is a polynomial of degree n in α whose coefficients are homogeneous of degree n in A. The case $\alpha = 1$ is the standard permanent (Minc, 1978), and $\alpha = -1$ is the determinant: $\mathrm{per}_{-1}(A) = \det(-A)$. The coefficient of degree one in α is the sum over cyclic permutations, i.e., permutations with $\mathrm{cyc}(\sigma) = 1$.

The α-determinant (Shirai, 2007) is defined in the same way with $\mathrm{cyc}(\sigma)$ replaced by the Cayley metric $n - \mathrm{cyc}(\sigma)$, which is the minimum number of factors required to represent σ as a product of transpositions. Thus $\mathrm{per}_\alpha(A) = \alpha^n \det_{1/\alpha}(A)$, and vice versa. The same algorithm may be used to compute both.

Vere-Jones (1997) gives a comprehensive account of the role of α-permanents in combinatorics, probability, statistics and quantum field theory. Diaconis, Graham and Holmes (2001) review the statistical applications of the standard permanent for binary matrices. The weighted permanent with positive half-integer α arises naturally in statistical work as the factorial moment

or product density for boson processes, and for negative α as the product density for fermion processes (Daley and Vere-Jones 2003; Shirai and Takahashi 2003; McCullagh and Møller 2006; Hough, Krishnapur, Peres and Virág 2006). The joint density at finite point configurations in these processes is expressed as an α-permanent, and the Papangelou conditional intensity is a ratio of α-permanents (Section 4). In such applications, the matrix A is invariably positive definite symmetric or Hermitian, sometimes strictly positive, with fairly large rank n, and usually $\alpha > 0$. For an application to classification, see McCullagh and Yang (2006)

The standard permanent is invariant under independent re-arrangement of rows and columns. In general, $\mathrm{per}_\alpha(HAH') = \mathrm{per}_\alpha(A)$ is invariant under *simultaneous* permutation of rows and columns.

Exact computation of the standard matrix permanent is known to be #P-complete (Valiant, 1979). Although the weighted permanent is not known to be #P-complete, the computational problem for general $\alpha \neq -1$ appears to be equally difficult. Jerrum, Sinclair and Vigoda (2004) have shown that the standard permanent of a non-negative matrix can be approximated with arbitrary accuracy in polynomial time of order $O(n^{10}(\log n)^3)$ by a Markov chain Monte Carlo algorithm, a bound subsequently reduced to $O(n^7(\log n)^4)$ by Bezáková, Stefankovic, Vazirani and Vigoda (2006). These papers demonstrate the existence of polynomial-time approximations: they could possibly be modified for general α, but they are at present not suitable for practical work, even for $\alpha = 1$. Alternative Monte Carlo algorithms with varying effectiveness are given by Beichl and Sullivan (1999) and Karmarkar *et al.* (1993) for binary matrices, and by Kuznetsov (1996) for $\alpha = 1$. Chen and Liu (2007) give an efficient sequential importance-sampling algorithm for binary matrices with $\alpha = 1$. Motivated by its marked efficacy in dealing with 0-1 matrices, this article proposes a sequential importance-sampling algorithm to estimate the α-weighted permanent for general α and generic matrices A. Our scheme is general but not foolproof; it is a practical algorithm designed to be efficient for symmetric positive definite matrices of the sort that arise in point process applications with $\alpha > 0$. Section 2 outlines the algorithm. Section 3 illustrates its efficiency through numerical examples. Section 4 discusses the estimation of ratio of permanents, a problem encountered in computing the Bayes estimate of the intensity function.

2. A SEQUENTIAL IMPORTANCE-SAMPLING ALGORITHM

To estimate the permanent using importance sampling, the basic idea is first to draw M permutations $\sigma_1, \ldots, \sigma_M$ independently from the set Π_n of permutations according to a probability distribution $f(\sigma)$, and then estimate the α-weighted permanent by $\sum_{i=1}^M w(\sigma_i)/M$, where

$$w(\sigma) = \frac{1}{f(\sigma)} \alpha^{\mathrm{cyc}(\sigma)} A_{1,\sigma(1)} \cdots A_{n,\sigma(n)}. \tag{1}$$

The random variables $w(\sigma_1), \ldots, w(\sigma_M)$ are independent and identically distributed with mean $\mathrm{per}_\alpha(A)$, so the variance of the average can be estimated in the usual way. The efficiency of an importance-sampling estimator depends crucially on the sampling distribution $f(\sigma)$. The naive choice of $f(\sigma) = 1/n!$, i.e., the uniform distribution over Π_n, is usually hopelessly inefficient. Following the classical result on importance sampling, we know that the optimal distribution is

$$f_{opt}(\sigma) \propto \alpha^{\mathrm{cyc}(\sigma)} |A_{1,\sigma(1)}| \cdots |A_{n,\sigma(n)}|, \tag{2}$$

which is not practically attainable, but does provide some hints of how to construct good sampling distributions.

We use sequential importance sampling (Liu, 2001) to build an approximation to $f_{opt}(\sigma)$ in successive steps. First, with a number k_1 chosen uniformly at random from $\{1, 2, \ldots, n\}$

we consider the possibilities for $\sigma(k_1)$. The optimal probability $\alpha^{\mathrm{cyc}(\sigma)}|A_{1,\sigma(1)}|\cdots|A_{n,\sigma(n)}|$ in (2) intuitively suggests that if $|A_{k_1,j}|$ is large compared with the other $|A_{k_1,j'}|$ ($j' \neq j$), then we should select $\sigma(k_1) = j$ with a high probability. However, letting the sampling probability $P(\sigma(k_1) = j)$ depend on the magnitude of $|A_{k_1,j}|$ alone brings only limited improvement over the naive choice of uniform selection. The reason is that once we select $\sigma(k_1) = j$, then effectively any other $\sigma(i)$, $i \neq k_1$, can no longer take the value j. Thus whether or not to assign $\sigma(k_1) = j$ should also be weighed against $\sigma(i) = j$ for $i \neq k_1$. As an illustration, suppose $\alpha = 1$ and $k_1 = 1$, and consider the 2×2 matrix

$$\begin{pmatrix} 100 & 2 \\ 1000 & 1 \end{pmatrix}. \tag{3}$$

If we take into consideration only the relative magnitude of $|A_{1j}|$ in the first row, we would assign $\sigma(1) = 1$ most of the time. But for this particular example choosing $\sigma(1) = 2$ with high probability makes the importance-sampling estimate much more efficient.

To incorporate the idea of not only comparing $|A_{k_1,j}|$ among themselves but also weighing $\sigma(k_1) = j$ against $\sigma(i) = j$ for $i \neq k_1$, and at the same time making the computation fast, we use a quick proxy. Let $C_j = \sum_{i=1}^{n} |A_{ij}|$ be the jth column sum of $|A|$. We assign $\sigma(k_1) = j$ with probability

$$p_j^{(1)} = \mathrm{pr}(\sigma(k_1) = j) \propto \alpha^{\Delta\,\mathrm{cyc}(\sigma(k_1)=j)} |A_{k_1,j}| / \left(C_j - |A_{k_1,j}|\right). \tag{4}$$

The ratio $|A_{k_1,j}|/(C_j - |A_{k_1,j}|)$ accounts for the relative importance of $\sigma(k_1) = j$ versus $\sigma(i) = j$ for some $i \neq k_1$. If this ratio is large for a particular j, we give $\sigma(k_1)$ a high chance to land on j. For the 2×2 matrix considered above, it is straightforward to check that the sampling rule (4) gives exactly the optimal weight (2). The notation $\Delta\,\mathrm{cyc}(\sigma(k_1) = j)$ in (4) denotes the number of *new* cycles formed by assigning $\sigma(k_1)$ to j. In other words, $\Delta\,\mathrm{cyc}(\sigma(k_1) = j) = 1$ if $j = k_1$ since $\sigma(k_1) = k_1$ creates the new cycle (k_1); and $\Delta\,\mathrm{cyc}(\sigma(k_1) = j) = 0$ otherwise. The general notation of $\Delta\,\mathrm{cyc}(\cdot)$ is used later in the discussion of ther algorithm. The factor $\alpha^{\Delta\,\mathrm{cyc}(\sigma(k_1)=j)}$ in (4) covers the case of general α; it is guided by the optimal weight (2).

Once $\sigma(k_1)$ is selected, we set all the matrix entries in row k_1 and column $\sigma(k_1)$ to zero, and update the column sums. This bookkeeping amounts to removing this row and column from further consideration. At the second step let $k_2 = \sigma(k_1)$ if $\sigma(k_1) \neq k_1$, and otherwise let k_2 be chosen uniformly from the remaining rows $\{1, \ldots, k_1 - 1, k_1 + 1, \ldots, n\}$. We then repeat the process on the reduced matrix beginning with row k_2. In other words, we track the open permutation cycle as it unfolds; otherwise, if the cycle is closed, we begin a new cycle by selecting a row uniformly from those that remain.

For row k_2, we choose among the available columns $\sigma(k_2) = j$ with probability

$$p_j^{(2)} = \mathrm{pr}(\sigma(k_2) = j) \propto \alpha^{\Delta\,\mathrm{cyc}(\sigma(k_2)=j)} |A_{k_2,j}| / \left(C_j^{(2)} - |A_{k_2,j}|\right), \tag{5}$$

where $C_j^{(2)}$ denotes the jth column sum of the matrix remaining at the second step, i.e. with row k_1 and column $\sigma(k_1)$ removed. The number of *new* cycles formed by assigning $\sigma(k_2)$ to j is $\Delta\,\mathrm{cyc}(\sigma(k_2) = j)$. For example, suppose $k_1 = 2$ and $\sigma(2) = 4$. Then the partial cycle $2 \to 4$ is completed if $\sigma(4) = 2$, so $\Delta\,\mathrm{cyc}(\sigma(4) = 2) = 1$; for all other $j \neq 2$, the sequence $2 \to 4 \to j$ is incomplete, so $\Delta\,\mathrm{cyc}(\sigma(4) = j) = 0$. Similarly, suppose $k_1 = 2$ and $\sigma(2) = 2$. Then assigning $\sigma(k_2) = k_2$ gives rise to a new cycle (k_2) yielding $\Delta\,\mathrm{cyc}(\sigma(k_2) = k_2) = 1$; for any other $j \neq k_2$, $\Delta\,\mathrm{cyc}(\sigma(k_2) = j) = 0$.

With $\sigma(k_2)$ chosen, we apply the same strategy to the third step: reset the entries in the k_2th row and the $\sigma(k_2)$-th column to zero, update the column sums to $C_j^{(3)}$, and move on to a new row k_3, which is row $\sigma(k_2)$ if it has not yet been considered, and otherwise is randomly chosen from the remaining available rows. We proceed in this sequential fashion until we reach the last available row and complete the final assignment. Each sampling step follows a probability rule parallel to (5).

The probability of getting all the assignments $\sigma(k_1), \sigma(k_2), \ldots, \sigma(k_n)$ is

$$p(k) = p^{(1)}_{\sigma(k_1)} p^{(2)}_{\sigma(k_2)} \cdots p^{(n-1)}_{\sigma(k_{n-1})}, \tag{6}$$

and

$$w(\sigma) = \frac{1}{p(k)} \alpha^{\mathrm{cyc}(\sigma)} A_{1,\sigma(1)} \cdots A_{n,\sigma(n)} \tag{7}$$

is an unbiased estimate of $\mathrm{per}_\alpha(A)$. The following pseudo-code summarizes the basic scheme of our algorithm.

Pseudo-code for the algorithm

For $m = 1, 2, \ldots, M$ /* generate M Monte Carlo samples in total */
  $C_j \leftarrow$ the jth column sum of $|A|$, $j = 1, \ldots, n$
  $R \leftarrow \{1, 2, \ldots, n\}$ /* columns still available */
  $k_1 \leftarrow$ chosen uniformly from R
  For $i = 1, \ldots, n-1$
    If any $C_j = 0$
      Set $w_m \leftarrow 0$, exit the current loop and move on to generate the next sample
    $p_j^{(i)} \leftarrow \alpha^{\Delta\,\mathrm{cyc}(\sigma(k_i)=j)} |A_{k_i,j}| / (C_j - |A_{k_i,j}|)$ for $j \in R$
    If (more than one $p_j^{(i)} = \infty$) or ($p_j^{(i)} = 0$ for all $j \in R$)
      Set $w_m \leftarrow 0$, exit the current loop and move on to generate the next sample
    Normalize $p_j^{(i)}$ such that $\sum_{j \in R} p_j^{(i)} = 1$
    Sample $\sigma(k_i)$ from R with probability $p_j^{(i)}$
    $R \leftarrow R \backslash \{\sigma(k_i)\}$ /* update the list of available columns */
    $C_j \leftarrow C_j - |A_{k_i,j}|$ for $j \in R$ /* update the column sum */
    If the assignment of $\sigma(k_i)$ completes the current cycle
      $k_{i+1} \leftarrow$ chosen uniformly from R
    else
      $k_{i+1} \leftarrow \sigma(k_i)$
  endfor
  $w_m \leftarrow \alpha^{\mathrm{cyc}(\sigma)} \prod_{i=1}^n \left(A_{k_i,\sigma(k_i)} / p^{(i)}_{\sigma(k_i)} \right)$
endfor

Two features of the algorithm make it efficient for practical computation: (i) tracking the formation of permutation cycles gives $\mathrm{cyc}(\sigma)$ as a useful by-product; (ii) as is common to sequential importance-sampling methods, the weight function $w(\sigma)$ in (7) can be computed recursively along the sampling steps: $w(\sigma) = \alpha^{\mathrm{cyc}(\sigma)} \prod_{i=1}^n \left(A_{k_i,\sigma(k_i)} / p^{(i)}_{\sigma(k_i)} \right)$.

A careful reader may notice that formula (7) does not exactly follow the importance-sampling rule (1), so its correctness is not obvious. The next theorem gives a proof.

THEOREM 1. *The ratio $w(\sigma)$ in (7) is an unbiased estimate of* $\mathrm{per}_\alpha(A)$.

Proof. Let $[n] = \{1, \ldots, n\}$ and let $\mathcal{K}_n$ be the set of ordered partitions of $[n]$. An ordered partition is an ordered list of disjoint non-empty ordered blocks or sub-lists whose union is $[n]$. For example, $324|61|5$, $5|61|432$ and $243|16|5$ are distinct ordered partitions of $[6]$, each having three blocks. There are $n!\,(n-1)!/((r-1)!\,(n-r)!)$ ordered partitions having r blocks, so the total number of ordered partitions of $[n]$ is $n!\,2^{n-1}$.

To each ordered partition $k \in \mathcal{K}_n$ there corresponds a permutation $s(k)$ of $[n]$, each block of the partition corresponding to a cycle, the order within blocks being maintained in the cycle. For example, the ordered partition $324|61|5$ generates the permutation $\sigma = (1,6)(2,4,3)(5)$ in standard cycle format. The order in which the cycles of σ are listed is immaterial, but the standard format lists each cycle beginning with the least element, and lists the cycles in increasing order of least element.

Using this notation, we re-write the weighted permanent as a sum over $\mathcal{K}_n$ in the form

$$\begin{aligned}\mathrm{per}_\alpha(A) &= \sum_{\sigma\in\Pi_n} \alpha^{\mathrm{cyc}(\sigma)} A(\sigma) = \sum_{\sigma\in\Pi_n} \alpha^{\mathrm{cyc}(\sigma)} A(\sigma) \sum_{\{k:s(k)=\sigma\}} q(k) \\ &= \sum_{k\in\mathcal{K}_n} \alpha^{\mathrm{cyc}\{s(k)\}} A\{s(k)\}\, q(k),\end{aligned}$$

where $A(\sigma) = \prod_j A_{j,\sigma(j)}$, and $q(\cdot)$ is any function such that $\sum_{\{k:s(k)=\sigma\}} q(k) = 1$ for all σ. This expression implies that if k is a random ordered partition whose distribution $g(\cdot)$ is strictly positive on $\mathcal{K}_n$, then

$$\alpha^{\mathrm{cyc}\{s(k)\}} A\{s(k)\} \frac{q(k)}{g(k)} \tag{8}$$

is an unbiased estimate of $\mathrm{per}_\alpha(A)$. One particular choice of $q(\cdot)$ is the following. For an ordered partition k consisting of r blocks of lengths $l_1, \ldots, l_r$, let

$$q(k) = 1/\{n(n-l_1)(n-l_1-l_2)\cdots(n-l_1-\cdots-l_{r-1})\}.$$

It is straightforward using induction to verify that $\sum_{\{k:s(k)=\sigma\}} q(k) = 1$ for all σ. In fact, our sequential algorithm generates ordered partitions, and the probability of obtaining a particular k is $g(k) = q(k)p(k)$ with $p(k)$ defined in (6). The cancellation of $q(k)$ in (8) results in the unbiased estimate (7). □

3. NUMERICAL ILLUSTRATIONS

To illustrate the algorithm, we first consider two 20×20 matrices A_1 and A_2. For the special case of $\alpha = 1$, 20×20 is close to the feasible limit for exact computation using the Ryser algorithm (van Lint and Wilson, 1992). Each component of A_1 is chosen uniformly and independently from $\{1, 2, \ldots, 10\}$; likewise, each entry of A_2 is chosen uniformly and independently from $\{0, 1, \ldots, 10\}$, so some entries of A_2 are zero. All matrices used in these tests are available on the authors' web pages `www.fas.harvard.edu/~skou/publication.htm` and `www.stat.uchicago.edu/~pmcc/reports`. For general $\alpha \neq \pm 1$, 15×15 is close to the feasible limit for exact computation of $\mathrm{per}_\alpha(A)$ on a PC. Using the algorithm, we estimated $\mathrm{per}_1(A_1)$ and $\mathrm{per}_1(A_2)$ as well as $\mathrm{per}_{1/2}(A_3)$ and $\mathrm{per}_{1/2}(A_4)$ for two randomly generated 15×15 matrices A_3 and A_4. Rows 1–4 of Table 1 summarize the Monte Carlo estimates.

Table 1. *Approximate and exact permanents for various matrices*

	Exact value	MC estimate†	Relative error (%)
$\mathrm{per}_1(A_1)$	9.7837×10^{32}	9.7871 ± 0.0136	0.14
$\mathrm{per}_1(A_2)$	3.5136×10^{32}	3.5058 ± 0.0074	0.21
$\mathrm{per}_{1/2}(A_3)$	1.4391×10^{22}	1.4372 ± 0.0032	0.22
$\mathrm{per}_{1/2}(A_4)$	7.0341×10^{21}	7.0433 ± 0.0223	0.32
$\mathrm{per}_1(A_5)$	3.2900×10^{49}	3.2939 ± 0.0124	0.38
$\mathrm{per}_1(A_6)$	5.9461×10^{40}	5.7816 ± 0.1025	1.72
$\mathrm{per}_{1/2}(A_7)$	2.0953×10^{31}	2.0924 ± 0.0078	0.37
$\mathrm{per}_{1/2}(A_8)$	1.5793×10^{25}	1.5494 ± 0.0265	1.68
$\mathrm{per}_{1/2}\{K(x)_9\}$	4.5051×10^{0}	4.5036 ± 0.0195	0.43
$\mathrm{per}_{1/2}\{K(x)_{11}\}$	1.6234×10^{2}	1.6223 ± 0.0091	0.56
$\mathrm{per}_{1/2}\{K(x)_{13}\}$	5.8158×10^{3}	5.8441 ± 0.0262	0.45
$\mathrm{per}_{1/2}\{K(x)_{15}\}$	2.1143×10^{5}	2.1174 ± 0.0112	0.53
$\mathrm{per}_{1/2}\{K(x)^{Tr}_{100}\}$	1.9109×10^{-16}	1.9282 ± 0.0363	1.90

† With standard error, and power of 10 as in column 2.

Each estimate was obtained from 20,000 independent Monte Carlo samples, i.e., $M = 20,000$ in (8), and took only *one second* on a 1.5 GHz Pentium PC. With the standard error less than 0.3% of the true value in all four cases, the algorithm appears to work well.

Following the suggestions of a referee, we consider more challenging matrices with *unbalanced* entries: A_5 is a 20×20 matrix whose (i, j) entry is sampled uniformly from $\{1, 2, \ldots, ij\}$, i.e., the product of row and column numbers. Likewise, A_6 is a 20×20 matrix similarly constructed, except that it is banded: the (i, j) entry is zero if $|i - j| > 3$. For $\alpha = 1/2$, we similarly generated two random 15×15 matrices A_7 and A_8: the (i, j) entry is uniform from 1 up to ij; A_8 is hepta-diagonal. Rows 5 to 8 of Table 1 compare the exact values of $\mathrm{per}_1(A_5)$, $\mathrm{per}_1(A_6)$, $\mathrm{per}_{1/2}(A_7)$ and $\mathrm{per}_{1/2}(A_8)$ with the approximation generated by the algorithm. Each estimate was again obtained from 20,000 Monte Carlo samples, taking one second on a PC. With small standard errors across all the cases, the algorithm is seen to work robustly well, though its estimate for banded matrices is not as precise as that for full non-banded matrices.

To examine how the algorithm performs as n increases, we apply it to a sequence of matrices with $n = 9, 11, 13, 15$ respectively. Each matrix has its (i, j) entry $\exp(-(x_i - x_j)^2)$, where $x = (x_1, \ldots, x_n)$ are independent Student t variables on five degrees of freedom. We use $K(x)_n$ to denote these matrices, viewing them as the covariance kernel evaluated at x. Rows 9 to 12 of Table 1 show that the Monte Carlo relative error is fairly constant at 0.5% over this range.

For the final example, we consider a matrix $K(x)_{100}$, whose (i, j) entry is $\exp\{-(x_i - x_j)^2/2\}$, where $x_1, \ldots, x_{100}$ are drawn independently from the symmetric triangular distribution on $(-\pi, \pi)$. See Fig. 2 for the location of these points. This size is far beyond the feasible limit for exact computation, so it is impossible to know the true value. From $M = 100,000$ Monte Carlo samples, we estimated $\mathrm{per}_{1/2}\{K(x)_{100}\}$ to be 2.706×10^{111} with relative error 0.9%, a computation that took only forty seconds on a 1.5 GHz Pentium PC. The histogram in Fig. 1 of the logarithm of the simulated values $\log_{10} w(\sigma_i)$ shows that the tail is well behaved. To check the accuracy of the algorithm for large matrices we truncate $K(x)_{100}$ to a tri-diagonal matrix $K(x)^{Tr}_{100}$, whose (i, j) entry is zero for all $|i - j| > 1$. The last row of Table 1 shows that the Monte Carlo estimate and standard error are consistent with the true value, which can be computed recursively for tri-diagonal matrices. The 1.9% relative error confirms the pattern

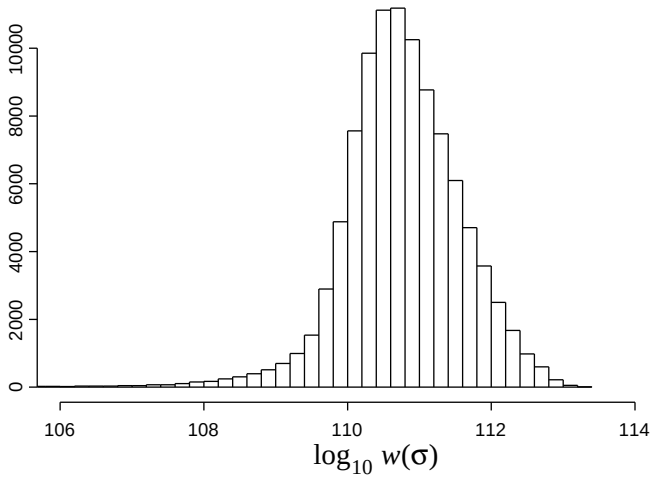

Fig. 1. The sampling distribution of the Monte Carlo outputs $\log_{10}\{w(\sigma_i)\}$ for the 100×100 matrix example.

established for smaller matrices, that the algorithm is less precise for banded matrices than for full matrices.

As we remarked in Section 1, the simulated annealing algorithms by Jerrum, Sinclair and Vigoda (2004) and by Bezáková, Stefankovic, Vazirani and Vigoda (2006) have good theoretical bounds, but at present they are not suitable for practical work, say $n \geq 25$, even for $\alpha = 1$. The algorithm by Karmarkar *et al.* (1993) is designed exclusively for 0-1 matrices with $\alpha = 1$; it generates random matrices related to A and uses the average of their determinants to estimate the permanent. Since the computation cost of this algorithm grows exponentially in n, it is at present not suitable for practical use even for $\alpha = 1$. The importance-sampling algorithm by Beichl and Sullivan (1999), designed for 0-1 matrices, uses Sinkhorn balance (Sinkhorn, 1964) to approximate the number of non-zero terms with particular matrix elements fixed. Since computing Sinkhorn balance is much more involved than simply computing a few ratios as in our algorithm, Beichl and Sullivan's algorithm requires much longer computing time. More importantly, since it is specifically designed for 0-1 matrices with $\alpha = 1$, it is not applicable for generic matrices with $\alpha \neq 1$. The algorithm by Kuznetsov (1996) uses sequential importance sampling. Starting from the first row, it samples $\sigma(1)$ with probability $\mathrm{pr}(\sigma(1) = j) = A_{1j} / \sum_{k=1}^{n} A_{1k}$, and then systematically moves to the second row, the third row and so on. The 2×2 matrix (3) illustrates the difficulty with this algorithm: 98% of time it assigns $\sigma(1) = 1$, which results in a poor estimate. The sequential importance-sampling algorithm by Chen and Liu (2007) starts by assigning $\sigma(1)$ and successively moves on to $\sigma(2)$, $\sigma(3)$, and so on. Since it is specifically designed for 0-1 matrices with $\alpha = 1$, it cannot be applied to generic matrices with $\alpha \neq 1$. In our experience, among all the algorithms constructed specifically for $\alpha = 1$ and 0-1 matrices, this algorithm works most efficiently. Its marked efficacy motivated our search for a similar algorithm to compute the α-permanent for general matrices.

4. RATIO OF PERMANENTS

In statistical problems connected with point processes, we are often interested in the ratio of permanents rather than their absolute magnitude. The simplest boson point process is a Cox process in which the intensity is either a squared zero-mean Gaussian process with covariance function $C/2$, or the sum of 2α independent squared Gaussian processes. On a bounded region S, the probability of observing $x = \{x_1, \ldots, x_n\}$ is proportional to $\mathrm{per}_\alpha\{K(x)\}$ where K is another covariance function related to C. The Papangelou conditional intensity at y is equal to the ratio $\mathrm{per}_\alpha\{K(x \cup y)\}/\mathrm{per}_\alpha\{K(x)\}$, where $K(x \cup y)$ is the matrix of order $n+1$ evaluated at the points $x \cup y$. The relation between K and C is described in Section 2.3 of McCullagh and Møller (2006). Both are positive definite and have the same eigenspaces, but the spectral norm of K is less than one, whereas the eigenvalues of C may be arbitrarily large. Usually $C(y, y')$ is non-negative, but there may be points at which $K(y, y') < 0$.

For a boson process, the permanental ratio $\mathrm{per}_\alpha\{K(x \cup y)\}/\mathrm{per}_\alpha\{K(x)\}$ is the conditional expectation of the intensity at y, given the observation $x \subset S$. Equivalently, if the squared Gaussian process is regarded as a prior distribution on intensity functions, the permanental ratio is the Bayes estimator of the intensity. For large α, this estimator is a kernel function

$$\alpha K(y, y) + \sum_i K(y, x_i)K(x_i, y)/K(x_i, x_i) + O(\alpha^{-1}),$$

that is, an additive function of the configuration x. For more typical values of α, the conditional intensity is not additive in x since it depends on the configuration of pairs, triples and so on. In general, if $K(y, y') \to 0$ for large $|y - y'|$, the conditional intensity for large $|y|$ is $\alpha K(y, y)$, so the conditional intensity is usually not integrable. For details, see McCullagh and Møller (2006).

In practical work, it is often more natural to consider a parametric family of boson processes, all of the same type associated with a family of covariance functions, say $\sigma^2 \exp(-|x - x'|/\tau)$ for $\sigma, \tau > 0$. In addition, α may be regarded as a parameter. The standard non-Bayesian procedure is to estimate the unknown parameter by maximum likelihood or cross-validation. This fitted process is then used to compute the conditional expected intensity at each point y using the permanental ratio. Since the density function for the boson model is proportional to $\mathrm{per}_\alpha\{K(x)\}$, the evaluation of α-permanents is also crucial for maximum likelihood estimation. A full Bayesian analysis is much more complicated because the prior is no longer a squared Gaussian process, but a mixture of such processes.

The naive method for computing $\mathrm{per}_\alpha\{K(x \cup y)\}/\mathrm{per}_\alpha\{K(x)\}$ is to estimate the numerator and denominator separately and compute the ratio. However, since $K(x \cup y)$ differs from $K(x)$ only by an extra row and column, we can take advantage of this special structure.

To each permutation σ of $[n]$ there correspond $n+1$ permutations σ' of $[n+1]$ generated by choosing an element i from $[n+1]$ and proceeding as follows. Set $\sigma'(i) = n+1$, $\sigma'(n+1) = \sigma(i)$ if $i \le n$, and for each $j \ne i$ set $\sigma'(j) = \sigma(j)$. For $i \le n$, the new element $n+1$ is inserted between i and $\sigma(i)$ in the cycle representation; otherwise the new element is a fixed point of σ'. The inverse operation $\sigma' \mapsto \sigma$ is the standard projection $\Pi_{n+1} \to \Pi_n$ on permutations that deletes element $n+1$ from the cycle representation.

With this understanding, we can write, using the shorthand notation $\mathrm{K} = K(x \cup y)$,

$$\mathrm{per}_\alpha(\mathrm{K}) = \sum_{\sigma \in \Pi_n} \alpha^{\mathrm{cyc}(\sigma)} \left(\prod_{i=1}^{n} \mathrm{K}_{i,\sigma(i)} \right) \left(\sum_{i=1}^{n} \frac{1}{\mathrm{K}_{i,\sigma(i)}} \mathrm{K}_{i,n+1}\mathrm{K}_{n+1,\sigma(i)} + \alpha \mathrm{K}_{n+1,n+1} \right).$$

In other words, the α-permanent of the $(n+1) \times (n+1)$ matrix K can be expressed in terms of permutations of $[n]$, which implies that after we sample an ordered partition $k \in \mathcal{K}_n$ with the

corresponding permutation $\sigma = s(k)$,

$$v(\sigma) = \frac{1}{p(k)} \alpha^{\mathrm{cyc}(\sigma)} \left(\prod_{i=1}^{n} \mathrm{K}_{i,\sigma(i)} \right) \left(\sum_{i=1}^{n} \frac{1}{\mathrm{K}_{i,\sigma(i)}} \mathrm{K}_{i,n+1} \mathrm{K}_{n+1,\sigma(i)} + \alpha \mathrm{K}_{n+1,n+1} \right), \quad (9)$$

with $p(k)$ given in (6), provides an unbiased estimate of $\mathrm{per}_\alpha(\mathrm{K})$. Thus, in the process of computing $\mathrm{per}_\alpha\{K(x)\}$ we automatically obtain an estimate for $\mathrm{per}_\alpha\{K(x \cup y)\}$. With M permutations $\sigma_i = s(k_i)$ obtained from M independent ordered partitions $k_1, \ldots, k_M$, the permanent ratio is estimated by

$$R = \left(\sum_{i=1}^{M} v(\sigma_i) \right) \Big/ \left(\sum_{i=1}^{M} w(\sigma_i) \right). \quad (10)$$

Furthermore, the intrinsic high correlation between $w(\sigma_i)$ and $v(\sigma_i)$ offers a big advantage for ratio estimation, which can be seen from the asymptotic variance

$$\mathrm{var}(R) \simeq \frac{1}{M} \frac{1}{\bar{w}^2} (R^2 s_w^2 + s_v^2 - 2R s_{wv}),$$

where $\bar{w}$ is the average of $w(\sigma_i)$, s_w^2, s_v^2 and s_{wv} are the sample variances and covariances of the pairs $w(\sigma_i), v(\sigma_i)$. A strong correlation between the w and v greatly reduces the variance of R. As a consequence, we can estimate $\mathrm{per}_\alpha\{K(x \cup y)\}/\,\mathrm{per}_\alpha\{K(x)\}$ accurately without having precise estimates of the individual terms.

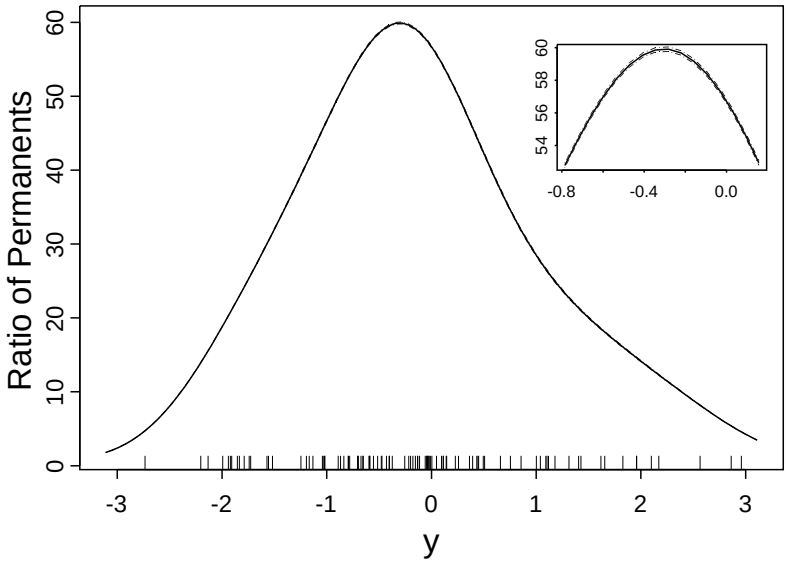

Fig. 2. The solid line shows the estimated ratio $\mathrm{per}_{1/2}\{K(x \cup y)\}/\,\mathrm{per}_{1/2}\{K(x)\}$ as a function of y. The dashed lines visible in the inset show the pointwise standard error. The points $x_1, \ldots, x_{100}$ are marked by the ticks along the horizontal axis.

In our last example $x = (x_1, \ldots, x_{100})$ consists of 100 points drawn independently from the triangular distribution on $(-\pi, \pi)$, and $K(x)$ is the covariance kernel $K(x, x') = \exp(-(x -$

$x')^2)$ evaluated at these points. Figure 2 shows the Monte Carlo estimate of the conditional intensity $\mathrm{per}_{1/2}\{K(x \cup y)\}/\,\mathrm{per}_{1/2}\{K(x)\}$ as a function of y. The sample points $x_1, \ldots, x_{100}$ are marked by the ticks along the horizontal axis.

The entire curve together with the error bars is computed from $M = 10,000$ Monte Carlo samples using (9) and (10); on a PC with a 1.5 GHz Pentium processor the whole computation took only *seven seconds*. The error bars in this case are so small that the dashed lines overlap with the solid line. The peak is magnified in the inset to show the error bars.

This analysis demonstrates that the ratio of permanents is easier to estimate than individual permanents, and the example shows that the algorithm is quite effective for estimating permanents that arise in point-process applications.

ACKNOWLEDGEMENTS

Support for this research was provided in part by the U.S. National Science Foundation. The authors thank Professor Jun Liu for helpful discussion.

REFERENCES

[19] Beichl, I. and Sullivan, F. (1999). Approximating the permanent via importance sampling with application to the dimer covering problem. *J. Computational Physics* **149**, 128–147.
[19] Bezáková, I., Stefankovic, D., Vazirani, V.V. and Vigoda, E. (2006). Accelerating simulated annealing for the permanent and combinatorial counting problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms* 900–907.
[19] Chen, Y. and Liu, J. S. (2007). Sequential Monte Carlo methods for permutation tests on truncated data. *Statistica Sinica* **17**, 857–872.
[19] Daley, D.J. and Vere-Jones, D. (2003). *An Introduction to the Theory of Point Processes* Vol. 1. *Elementary Theory and Methods* 2nd edn. Springer, New York.
[19] Diaconis, P., Graham, R.L. and Holmes, S. (2001). Statistical problems involving permutation with restricted positions. *State of the Art in Probability and Statistics*. Ed. M. de Gunst, C. Klaassen and A. Van der Vaart, pp. 195–222. Institute of Mathematical Statistics, Hayward, CA.
[19] Hough, J., Krishnapur, M., Peres, Y. and Virág, B. (2006). Determinantal processes and independence. *Probability Surveys* **3**, 206–229.
[19] Jerrum, M.R., Sinclair, A. and Vigoda, E. (2004). A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries. *J. Assoc. Comput. Mach.* **51**, 671–697.
[19] Karmarkar, N., Karp, R., Lipton, R., Lovász, L. and Luby, M. (1993). A Monte Carlo algorithm for estimating the permanent. *SIAM J. Comput.* **22**, 284–293.
[19] Kuznetsov, N. (1996). Computing the permanent by importance sampling method. *Cybernetics and System Analysis* **32**, 749–755.
[19] Liu, J.S. (2001). *Monte Carlo Strategies in Scientific Computing*. Springer, New York.
[19] McCullagh, P., and Møller, J. (2006). The permanental process. *Adv. Appl. Prob.* **38**, 873–888.
[19] McCullagh, P. and Yang. J. (2006). Stochastic classification models. *Proc. Int. Congress of Mathematicians, vol III*. Ed. M. Sanz-Solé, J. Soria, J.L. Varona and J. Verdera, pp. 669–686. Zurich: European Mathematical Society Publishing House.
[19] Minc, H. (1978). *Permanents*. Encyclopedia of Mathematics and its Applications vol. 6. Addison-Wesley, Reading, MA.
[19] Shirai, T. and Takahashi, Y. (2003) Random point fields associated with certain Fredholm determinants I: fermion, Poisson and boson point processes. *J. Functional Analysis* **205**, 414–463.
[19] Shirai, T. (2007) Remarks on the positivity of α-determinants. *Kyushu J. Math.* **61**, 169–189.
[19] Sinkhorn, R. (1964) A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist.* **35**, 876–879.
[19] Valiant, L.G. (1979). The complexity of computing the permanent. *Theoretical Computer Science* **8**, 189–201.
[19] van Lint, J., and Wilson, R. (1992). *A Course in Combinatorics*. Cambridge University Press, Cambridge, UK.
[19] Vere-Jones, D. (1997) Alpha-permanents and their application to multivariate gamma, negative binomial and ordinary binomial distributions. *New Zealand J. Math.* **26**, 125–149.

[*Received April* 2008. *Revised August* 2008]