A Faster Practical Approximation Scheme for the Permanent

Juha Harviainen, Mikko Koivisto

University of Helsinki
juha.harviainen@helsinki.fi, mikko.koivisto@helsinki.fi

Abstract

The permanent of a matrix has numerous applications but is notoriously hard to compute. While nonnegative matrices admit polynomial approximation schemes based on rapidly mixing Markov chains, the known practical estimators of the permanent rely on importance or rejection sampling. We advance the rejection sampling approach, which provides probabilistic accuracy guarantees, unlike importance sampling. Specifically, we give a novel class of nesting upper bounds and a simple preprocessing method that, in comparison to previous works, enable faster sampling with better acceptance rate; we demonstrate order-of-magnitude improvements with both theoretical and empirical analyses. In addition, we display instances on which our approximation scheme is competitive against state-of-the-art importance sampling based estimators.

Introduction

The permanent of an $n \times n$ matrix $A = (a_{i,j})$ is the sum

$$\operatorname{per} A \coloneqq \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

over all permutations σ of $\{1, 2, \ldots, n\}$. Permanents appear in numerous and diverse applications, such as counting perfect matchings in a bipartite graph, helping to verify a quantum-mechanical formula (Aaronson and Arkhipov 2011), heuristic counting under global cardinality constraints (Bianco et al. 2019), and multi-target tracking (Uhlmann 2004), to name just a few.

For computing the permanent, the fastest known exact algorithms run in time $O(2^n n)$ (Ryser 1963; Nijenhuis and Wilf 1978), and no polynomial-time algorithm presumably exists, for the problem is #P-hard (Valiant 1979). As even deciding the sign of the permanent is #P-hard (Jerrum, Sinclair, and Vigoda 2004), research on approximation algorithms has focused on nonnegative matrices, which suffice for many applications. Nonnegative matrices admit fully polynomial randomized approximation schemes based on rapidly mixing Markov chains (Jerrum, Sinclair, and Vigoda 2004). However, the best known time complexity bounds are impractical, within polylogarithmic factors of n^7 (Bezáková et al. 2008), and there is also empirical evidence of infeasibility (Newman and Vardi 2020).

The practical intractability of the problem calls for approaches developed mainly in the fields of AI and operations research. Instead of requiring tractability in the worst case (or average case over some simplistic baseline distribution), the goal is to design methods that make use of *heuristics* so as to perform well on various real-world and benchmark instances. While this paradigm has generally been successful for (weighted) model counting (see, e.g., Gomes, Sabharwal, and Selman 2006, Soos and Meel 2019, Kuck et al. 2019b), the available generic methods appear to be inefficient for approximating the permanent. Thus, estimating the permanent of nonnegative matrices—in addition to being a fundamentally important problem on its own—serves as a concrete hard benchmark for heuristic-driven methods.

In this paper, we advance a rejection sampling approach for estimating the permanent, put recently forward by Kuck et al. (2019a) and Harviainen, Röyskö, and Koivisto (2021). The approach stems from Huber's (2006) scheme for binary matrices. The idea is to sample a uniform random value between 0 and some upper bound $U(A)$ of the permanent; if the value falls in a range of size $w(\sigma) := \prod_{i=1}^{n} a_{i,\sigma(i)}$ reserved by some permutation σ, then the sample is accepted, otherwise rejected. The acceptance rate is $\operatorname{per} A / U(A)$, and after sufficiently many accepts, we obtain a high-confidence estimate of $\operatorname{per} A$. This approach relies on two key conditions: (i) we can efficiently decide whether to accept or reject a sample; (ii) we have a good bound $U(A)$. To fullfill the first condition, Huber utilized self-reducibility: a partial permutation is incrementally extended until it gets accepted or rejected—crucially, this only works if the bound $U(A)$ is *nesting*, a kind of monotonicity property. This requirement prevented using the well-known Minc–Brègman bound (Minc 1963; Brègman 1973). As a remedy, Huber crafted an approximate variant of that bound. Huber and Law (2008) extended the idea to matrices with arbitrary nonnegative real entries.

Kuck et al. (2019a) circumvented the need for a nesting bound. Their AdaPart scheme uses the tighter Minc–Brègman bound, or its extension to nonbinary entries (Schrijver 1978; Soules 2005), and adaptively refines the partition of the sampling space until the required monotonicity condition is satisfied. A drawback is the larger computational

Name	Domain	Row bound	Nesting	References	Comment
Minc–Brègman	binary	$\gamma(r_i) := (r_i!)^{1/r_i}$	No	(Minc 1963; Brègman 1973)	Tightest known
Brouwer–Schrijver	real	$\sum_{k=1}^{n}(\gamma(k)-\gamma(k-1))a^*_{i,k}$	No	(Schrijver 1978; Soules 2005)	Extends Minc–Brègman
Huber	binary	$h(r_i) := g(r_i)/\mathrm{e}$, Eq. (1)	Yes	(Huber 2006)	Relaxes Minc–Brègman
Huber–Law	real	$\ell(r_i)/\mathrm{e}$, Eq. (3)	Yes	(Huber and Law 2008)	Similar to Huber
Extended Huber	real	$\sum_{k=1}^{n}(h(k)-h(k-1))a^*_{i,k}$	Yes	*this paper*	Extends Huber

Table 1: Upper bounds for the permanent of a nonnegative $n \times n$ matrix $(a_{i,j})$, the sum and kth largest entry of row i denoted by r_i and $a^*_{i,k}$

cost per sample, traded for the higher acceptance rate, which renders the scheme slower than the one by Huber and Law (2008) for some classes of matrices (Harviainen et al. 2021).

The main contribution of the present work is a new nesting upper bound for the permanent of nonnegative real matrices. Our key observation is that the decomposition technique used for extending the Minc–Brègman bound *preserves the nesting property*.[1] Thus, by taking any nesting bound for binary matrices and extending it using the technique, we obtain a nesting bound for nonnegative real matrices. We do exactly this with Huber's nesting bound; Table 1 summarizes the mentioned bounds. We give both analytical and empirical results showing that the resulting scheme is superior to the previous schemes by Huber and Law (2008) and by Kuck et al. (2019a). Since our new bound—like the other mentioned bounds—is a product of row bounds, the deep rejection sampling method of Harviainen et al. (2021) applies to our new bound as well and boosts the scheme further. Another way to enhance performance is to appropriately transform the input matrix as preprocessing. We revisit *sharpening*, a simple transformation due to Soules (2005), also mentioned but not used by Kuck et al. (2019a). We give a heuristic implementation of the transformation and find empirically that it usually significantly tightens the bound.

If one is ready to abandon accuracy guarantees, also other Monte Carlo estimators become available. In particular, importance sampling (IS) based estimators have appeared to often perform well in practice (Smith and Dawkins 2001; Alimohammadi et al. 2021). Even if the number of samples required for accuracy guarantees is generally too high to be useful, recent advances in sample complexity analysis (Chatterjee and Diaconis 2018) have enabled discovering classes of matrices for which the complexity is provably feasible (Alimohammadi et al. 2021). Could the state-of-the-art IS estimators be systematically more efficient than the best rejection sampling schemes (given in this paper), and we just do not know how to prove good sample complexity bounds? Or, does there exist (classes of) matrices on which IS estimators actually perform worse than our rejection sampling scheme, thus revealing a reason why proving good bounds is, in fact, impossible? We conducted a small-scale empirical study of this question and found evidence for the latter stand. It should be stressed that the comparison of IS and rejection sampling schemes is, of course, not fair, since only the latter provide approximation guarantees—this partly explains why the two methods have not been compared against each other in previous works.

While this work focuses on estimating the permanent, the questions studied here should also be considered in a broader context of approximate model counting. We will discuss this aspect in the last section of this paper, for instance, viewing the technique to extend a bound from binary matrices to nonnegative matrices as a way to "reduce" weighted to unweighted model counting (Chakraborty et al. 2015).

Rejection Sampling for Estimating the Permanent

We introduce Huber's (2006) rejection sampling scheme and necessary notation needed in the rest of the paper.

A map U from a set of matrices $\mathcal{S}$ to nonnegative reals is a *product upper bound in* $\mathcal{S}$, or just *product bound*, if for all $A \in \mathcal{S}$ we have $U(A) \geq \operatorname{per} A$ and

$$U(A) = \prod_{i=1}^{n} u\Big(\sum_{j=1}^{n} a_{i,j}\Big)$$

with some fixed function u that we call the *row bound*.

Denote by $f(A, i, j)$ the matrix obtained from A by removing the ith row and the jth column.

Definition 1. *A map U from a set of matrices $\mathcal{S}$ to nonnegative reals is* nesting *if*

$$U(A) \geq \sum_{i=1}^{n} a_{i,1} \cdot U(f(A, i, 1)) \quad \textit{for all } A \in \mathcal{S}\,.$$

Observe that a nesting map is necessarily an upper bound for the permanent. For generalized notions of nesting to arbitrary recursive partitioning of sampling spaces, see Huber (2006), Kuck et al. (2019a), and Harviainen et al. (2021).

Nesting bounds are desired as they enable exact sampling of permutations σ proportionally to the weights $w(\sigma)$ obtained as the product of the corresponding entries. Algorithm 1, due to Huber (2006), describes the $O(n^2)$-time self-reducible rejection sampling method for nesting bounds. When a draw is accepted, the probability that the computation path corresponds to σ equals $w(\sigma)/\operatorname{per} A$, yielding the total acceptance probability of $\operatorname{per} A/U(A)$: the tighter the bound, the higher the acceptance rate. Acceptance can also be understood as a Bernoulli-distributed random variable.

[1] Soules (2005) gives a generic treatment of this technique along with historical notes, but he does not consider the nesting property.

Algorithm 1: Nesting rejection sampling

Input : A nonnegative $n \times n$ matrix A, an upper bound U.
Output : Either *accept* or *reject*.
if A is an 1×1 matrix **then**
  return *accept*;
$b_i \leftarrow a_{i,1} U(f(A, i, 1))$ for every $i \in \{1, 2, \ldots, n\}$;
$s \leftarrow \sum_{i=1}^{n} b_i$;
Define $p(i) \coloneqq b_i / U(A)$ and $p(\mathit{reject}) \coloneqq 1 - s/U(A)$;
Sample X by using p as a probability mass function;
if $X = \mathit{reject}$ **then**
  return *reject*;
Call Algorithm 1 with parameters $f(A, X, 1)$ and U;
return the result from recursion;

The ratio of the number of accepted draws and the total number of draws multiplied by $U(A)$ gives an approximation of $\operatorname{per} A$. A result of Dagum et al. (2000) states that the estimate is an (ϵ, δ)-*approximation*, that is, its relative error is at most ϵ with probability at least $1-\delta$, as soon as the number of accepted draws exceeds $1+4(\mathrm{e}-2)(1+\epsilon)\epsilon^{-2}\ln(2/\delta)$. The number of required accepted draws can be slightly lowered by using Huber's (2017) GBAS algorithm.

Two row bounds for binary matrices are given more focus in the paper. The *Minc–Brègman bound* U_{MB} (Minc 1963; Brègman 1973) whose row bound is defined by $\gamma(0) \coloneqq 0$ and $\gamma(k) \coloneqq (k!)^{1/k}$ is the tightest possible product bound for $(0,1)$-matrices (Minc 1984). However, the bound is not nesting so the previous algorithm cannot be employed with it. The *Huber bound* U_{H} (Huber 2006) is its tightest known nesting relaxation. It uses $h(k) \coloneqq g(k)/\mathrm{e}$ as its row bound, with $g(k)$ given by the recurrence $g(0) \coloneqq 0, g(1) \coloneqq \mathrm{e}$, and

$$g(k+1) \coloneqq g(k) + 1 + \frac{1}{2g(k)} + \frac{0.6}{g(k)^2}. \qquad (1)$$

The Minc–Brègman bound and its generalizations require a more complicated procedure. The adaptive partitioning scheme (AdaPart) (Kuck et al. 2019a) maintains a set of matrices and a weighted sum of their upper bounds, and as long as this sum exceeds the bound of the current matrix, a random matrix A' from the set is replaced by matrices $f(A', 1, j), f(A', 2, j), \ldots, f(A', n, j)$ for some j. When the sum is at most the current bound, the bounds in the set are used as a probability mass function in a similar manner as in Algorithm 1. AdaPart requires $O(n^3)$ time assuming that $O(1)$ rounds of replacement always suffice.

New Nesting Product Bounds

We will show that when one takes Soules's (2005) decomposition extension of a nesting product bound in the set of $(0,1)$-matrices $\mathcal{B}$, then the resulting bound is nesting in the set of nonnegative matrices $\mathcal{A}$. Furthermore, we show that the extension of the Huber bound is competitive against the tighest known (non-nesting) product bound in $\mathcal{A}$ and tighter than the previous best nesting bound. We have deferred proofs of some lemmas (marked by †) to the supplemental material[2].

[2]https://github.com/Kalakuh/nesting

Decomposition Extensions

The permanent is a row-multilinear function, meaning that if matrices A, B, and C differ only on the ith row such that $A_i = B_i + C_i$, then $\operatorname{per} A = \operatorname{per} B + \operatorname{per} C$. Soules (2005) used this property to show that for any product bound U in $\mathcal{B}$, there is an *extended product bound* U^* in $\mathcal{A}$ defined as

$$U^*(A) \coloneqq \prod_{i=1}^{n} \sum_{k=1}^{n} \big(u(k) - u(k-1)\big) a^*_{i,k}$$

where $a^*_{i,k}$ is the kth largest entry on the ith row. The bound U^* reduces to U for matrices in $\mathcal{B}$. For example, the extended Minc–Brègman bound is the *Brouwer–Schrijver bound*, observed by Brouwer and described by Schrijver (1978), which AdaPart uses. Extended bounds enable rejection sampling for nonnegative matrices and preprocessing methods like Sinkhorn balancing (Sinkhorn 1964; Soules 1991) that alter the original matrix into a nonbinary one.

The bounds are based on decomposing $A \in \mathcal{A}$ into a conical combination of binary matrices in $\mathcal{B}$ such that the permanent of A is obtained as $\sum_{B \in \mathcal{B}} \theta_B \operatorname{per} B$ with $\theta_B \geq 0$. Then, applying an upper bound U on each B yields a new upper bound for the permanent of A. This idea is at the heart of our main result as well. Soules (2005) also showed that U^* is the tightest decomposition bound obtainable from U.

Extensions Preserve Nestedness

We start by proving our main result:

Theorem 1. *If U is a nesting product bound in $\mathcal{B}$, its decomposition extension U^* is a nesting product bound in $\mathcal{A}$.*

Proof. The idea is to write U^* as a conical combination of the bound U applied on $(0,1)$-matrices, following Soules (2005). Then, we argue that since U is nesting, so too is U^*.

Consider a single row vector A_i of matrix A. Letting $b_{i,k}$ be an n-dimensional $(0,1)$-vector with 1s only in the positions where A_i has its k largest entries, we can write A_i as

$$\sum_{k=1}^{n} (a^*_{i,k} - a^*_{i,k+1})\, b_{i,k}$$

with $a^*_{i,n+1} = 0$. Applying the row bound u on each $b_{i,k}$ yields, in a sense, a new row bound for the row vector:

$$\sum_{k=1}^{n} (a^*_{i,k} - a^*_{i,k+1})\, u(k) = \sum_{k=1}^{n} \big(u(k) - u(k-1)\big) a^*_{i,k}\,. \quad (2)$$

Similarly, let $B^{\boldsymbol{k}}$, with $\boldsymbol{k} = (k_1, k_2, \ldots, k_n)$, denote the $(0,1)$-matrix $(b^{\boldsymbol{k}}_{i,j})$ where the ith row has 1s in the positions of k_i largest entries on the ith row of $A \in \mathcal{A}$. Also, let $[n] \coloneqq \{1, 2, \ldots, n\}$. By previous formulas and row-multilinearity,

$$U^*(A) = \sum_{\boldsymbol{k} \in [n]^n} \overbrace{\left(\prod_{t=1}^{n} a^*_{t,k_t} - a^*_{t,k_t+1} \right)}^{P(\boldsymbol{k})} U(B^{\boldsymbol{k}})\,.$$

Since U is nesting, we have

$$U(B^{\boldsymbol{k}}) \geq \sum_{i=1}^{n} b^{\boldsymbol{k}}_{i,1} \cdot U\left(f(B^{\boldsymbol{k}}, i, 1)\right).$$

Combining and rearranging gives us that

$$U^*(A) \geq \sum_{i=1}^{n} \sum_{\boldsymbol{k}\in[n]^n} P(\boldsymbol{k}) \cdot b^{\boldsymbol{k}}_{i,1} \cdot U\big(f(B^{\boldsymbol{k}}, i, 1)\big).$$

To complete the proof, we fix i and show that

$$a_{i,1} \cdot U^*(f(A,i,1)) = \sum_{\boldsymbol{k}\in[n]^n} P(\boldsymbol{k}) \cdot b^{\boldsymbol{k}}_{i,1} \cdot U\big(f(B^{\boldsymbol{k}}, i, 1)\big).$$

To this end, let s_i denote the rank of the first entry on row i, i.e., $a^*_{i,s_i} = a_{i,1}$. Let $[Q]$ equal 1 when Q is true, and 0 otherwise. Now the right-hand side can be written as

$$\sum_{\boldsymbol{k}\in[n]^n} P(\boldsymbol{k}) \cdot [k_i \geq s_i] \cdot \prod_{\substack{t=1\\t\neq i}}^{n} u\big(k_t - [k_t \geq s_t]\big),$$

which can be rearranged to

$$\sum_{k_i=s_i}^{n} (a^*_{i,k_i} - a^*_{i,k_i+1}) \prod_{\substack{t=1\\t\neq i}}^{n} \sum_{j=1}^{n} (a^*_{t,j} - a^*_{t,j+1}) u(j - [j \geq s_t]).$$

By rearranging the summands like in Eq. (2), we see that the product equals $U^*(f(A,i,1))$ by the definition of extended bounds, and thus the expression simplifies to

$$\sum_{k_i=s_i}^{n} (a^*_{i,k_i} - a^*_{i,k_i+1})\, U^*(f(A,i,1))$$

and further to $a^*_{i,s_i}\, U^*(f(A,i,1)) = a_{i,1}\, U^*(f(A,i,1))$. This completes the proof. □

Extended Huber in the Worst Case

How bad can the extended Huber bound be in relation to the Brouwer–Schrijver bound? To address this question, we need the following lemma that follows from a bound for U_{H}, Stirling's formula, and the inequality $1 + x < \mathrm{e}^x$.

Lemma 2 (†)**.** *For all $k > 0$, $h(k) - \gamma(k) < 0.264$.*

We remark the constant could probably be lowered; namely, our calculations (omitted here) have shown that $h(k) - \gamma(k) < 0.194$ for all $k \leq 10^8$ and the value seems to converge rather quickly. The current bound leads to the following worst-case analysis:

Theorem 3. *For any nonnegative $n \times n$ matrix A, $U^*_H(A)/U^*_{MB}(A) \leq (h(3)/\gamma(3))^n < 1.042^n$.*

Proof. Consider a row with entries $a_1 \geq a_2 \geq \cdots \geq a_n$ and let $a_{n+1} = 0$. By applying Lemma 2, we get that

$$\frac{h(k)}{\gamma(k)} = 1 + \frac{h(k) - \gamma(k)}{\gamma(k)} < 1 + \frac{0.264}{\gamma(k)}.$$

As γ is an increasing function, we can show that $h(k)/\gamma(k) \leq h(3)/\gamma(3)$ for all $k > 0$ by manually checking its values until $1 + 0.264/\gamma(k) \leq h(3)/\gamma(3)$.

We can now bound the ratio of the bounds for a row:

$$\frac{\sum_{k=1}^{n}(a_k - a_{k+1})h(k)}{\sum_{k=1}^{n}(a_k - a_{k+1})\gamma(k)} \leq \frac{h(3)}{\gamma(3)},$$

and thus in the worst case $U^*_{\mathrm{H}}(A)/U^*_{\mathrm{MB}}(A) \leq (h(3)/\gamma(3))^n$.

This bound is tight for matrices that have exactly three identical entries on each row, the rest being zeros. □

Extended Huber in the Average Case

We next show that the extended Huber bound performs well with high probability for random $n \times n$ matrices A, in which the entries are independent and uniformly distributed in the interval $[0, 1]$. For simplicity, we write U^* for $U^*(A)$, understanding that U^* is a random variable through A.

We start by analyzing the expected value and the variance of U^* under a very mild assumption regarding the underlying row bound u: we assume that $\Delta_k := u(k) - u(k-1)$ is at most 1. All bounds in Table 1 satisfy this. We will only use the notation Δ_k when the bound is clear from the context.

The proofs of the following lemmas are based on the fact that the order statistics for independent uniformly distributed entries follow a Beta distribution.

Lemma 4 (†)**.** *We have*

$$\mathbf{E}[U^*] = \left(\frac{\sum_{k=1}^{n} u(k)}{n+1}\right)^n \geq \left(\frac{n}{2\mathrm{e}}\right)^n.$$

Lemma 5 (†)**.** *Suppose $\Delta_k \leq 1$ for all $k \geq 1$. Then $\mathrm{Var}[U^*] \leq (n/4)^n$.*

We get that the bound is arbitrarily close to the expected value for random matrices with high probability as n grows:

Proposition 6. *Suppose $\Delta_k \leq 1$ for all $k \geq 1$. Let $\epsilon > 0$. Then $|U^* - \mathbf{E}[U^*]| \leq \epsilon \cdot \mathbf{E}[U^*]$ with high probability.*

Proof. By Chebyshev's inequality and Lemmas 4 and 5,

$$\begin{aligned}\Pr\left(|U^* - \mathbf{E}[U^*]| \geq \epsilon \cdot \mathbf{E}[U^*]\right) &\leq \frac{\mathrm{Var}[U^*]}{\epsilon^2 \mathbf{E}[U^*]^2} \\ &\leq \frac{(2\mathrm{e})^{2n} n^n}{\epsilon^2 (2n)^{2n}} \\ &= \epsilon^{-2}(\mathrm{e}^2/n)^n,\end{aligned}$$

which tends to 0 as n tends to infinity. □

We now apply the above general results to the extended Huber bound and get that w.h.p. the bound is within a small constant factor of the Brouwer–Schrijver bound U^*_{MB}:

Theorem 7. *With high probability, $U^*_H/U^*_{MB} \leq 4.201$.*

Proof. We begin by noting that $U^*_{\mathrm{H}}/U^*_{\mathrm{MB}}$ is close to $\mathbf{E}[U^*_{\mathrm{H}}]/\mathbf{E}[U^*_{\mathrm{MB}}]$ with high probability. This follows by applying Proposition 6 to both U^*_{H} and U^*_{MB}. (The verification of the condition on Δ_k for both bounds is left to the reader.)

It remains to bound $\mathbf{E}[U^*_{\mathrm{H}}]/\mathbf{E}[U^*_{\mathrm{MB}}]$ from above. From Lemma 4 we get

$$\frac{\mathbf{E}[U^*_{\mathrm{H}}]}{\mathbf{E}[U^*_{\mathrm{MB}}]} = \left(\frac{\sum_{k=1}^{n} h(k)}{\sum_{k=1}^{n} \gamma(k)}\right)^n = \left(1 + \frac{\sum_{k=1}^{n} h(k) - \gamma(k)}{\sum_{k=1}^{n} \gamma(k)}\right)^n.$$

Lemma 2 and Stirling's inequality yield a strict upper bound

$$\begin{aligned}\left(1 + \frac{0.264n}{\sum_{k=1}^{n}(2\pi k)^{1/(2k)} k/\mathrm{e}}\right)^n &< \left(1 + \frac{0.264\mathrm{e}n}{\sum_{k=1}^{n} k}\right)^n \\ &\leq \left(1 + \frac{0.528\mathrm{e}}{n}\right)^n.\end{aligned}$$

Using $1 + x \leq \mathrm{e}^x$ gives the bound $\exp(0.528\mathrm{e}) < 4.201$. □

Extended Huber is Tighter Than Huber–Law

Finally, we show that the extended Huber bound is tighter than the *Huber–Law bound* U_{HL} (Huber and Law 2008), which is defined for matrices with entries from the interval $[0, 1]$ by a row bound $\ell(k)/\mathrm{e}$ with

$$\ell(k) := \begin{cases} k + (1/2)\ln k + \mathrm{e} - 1\,, & k \geq 1, \\ 1 + (\mathrm{e}-1)k\,, & k \in (0,1), \\ 0\,, & k = 0. \end{cases} \tag{3}$$

Any nonnegative matrix can be scaled so that its entries are in $[0, 1]$, so this requirement is not limiting the bound.

Lemma 8 (†). *For all matrices $B \in \mathcal{B}$, $U_{\mathrm{H}}(B) \leq U_{\mathrm{HL}}(B)$.*

Theorem 9. *For all matrices A with entries in $[0, 1]$, $U^*_{\mathrm{H}}(A) \leq U_{\mathrm{HL}}(A)$.*

Proof. We show first that $U^*_{\mathrm{HL}}(A) \leq U_{\mathrm{HL}}(A)$ by proving that

$$\sum_{k=1}^{n} (\ell(k) - \ell(k-1))\, a_k \leq \ell\Bigg(\sum_{k=1}^{n} a_k\Bigg)$$

for any decreasing sequence $a_1, a_2, \ldots, a_n \in [0, 1]$.

Denote $s_m := \sum_{k=1}^{m} a_k$ with $s_0 = 0$. Start by rewriting the right-hand side:

$$\ell(s_n) = \sum_{k=1}^{n} \ell(s_k) - \ell(s_{k-1})\,.$$

Because ℓ is concave, $\ell(x) - \ell(x - c)$ is decreasing in x for any fixed $c \in [0, 1]$, and thus

$$\begin{aligned} \ell(s_k) - \ell(s_{k-1}) &= \ell(s_{k-1} + a_k) - \ell(s_{k-1}) \\ &\geq \ell(k - 1 + a_k) - \ell(k-1)\,. \end{aligned}$$

Using the concavity of ℓ again gives us

$$\begin{aligned} &\ell(k - 1 + a_k) - \ell(k-1) \\ &\quad\geq \ell(k-1) + (\ell(k) - \ell(k-1))a_k - \ell(k-1) \\ &\quad= (\ell(k) - \ell(k-1))a_k, \end{aligned}$$

and thus $U_{\mathrm{HL}}(A) \geq U^*_{\mathrm{HL}}(A)$.

Any extended bound $U^*(A)$ is a conical combination of bounds $U(B)$ for its decomposition, so the inequality of Lemma 8 implies $U^*_{\mathrm{HL}}(A) \geq U^*_{\mathrm{H}}(A)$. □

Deep Bounds

The deep rejection sampling method by Harviainen et al. (2021) is applicable for the extended bounds. A depth-d variant of an upper bound U is the sum over all $d \times d$ submatrices A' of the first d columns of A where the summands are products of the permanent of A' and the bound U applied on the remaining submatrix in the last $n - d$ columns. The bound can be computed somewhat efficiently with $O(2^d dn)$ arithmetic operations for product bounds.

Deep rejection sampling starts by fixing entries in the first d columns using stochastic backtracking on a precomputed dynamic programming table, and then utilizes basic self-reducible rejection sampling applicable for U on the remaining rows and columns to complete the sample. The method can improve the running time by several orders of magnitude (Harviainen et al. 2021).

Preprocessing

Upper bounds for the permanent can sometimes be tightened by performing operations on the matrix and analyzing how they affect its permanent. For example, multiplying some column of the matrix by $x > 0$ multiplies the permanent by x as well, but the upper bound might increase by a factor that is less than x. Soules (2005) gives an overview of similar *sharpening* methods for the bounds.

Harviainen et al. (2021) experimented with the following version of Huber and Law's (2008) preprocessing method: First, entries not part of any permutation of positive weight are replaced by 0s with Tassa's algorithm (2012). Then, the matrix is transformed to nearly doubly stochastic by alternatingly dividing each row and each column by the sums of their entries n^2 times, as suggested by Sullivan and Beichl (2014). Finally, each row vector is divided by its largest entry to make its largest entry equal 1. We will refer to schemes that use this preprocessing by adding "-D" to their names.

Here, we propose a simple sharpening method for extended bounds: pick a random column n^2 times, multiply it by 2^x with $x \sim \mathrm{Uniform}[-1, 1]$, and check if the upper bound increases by a factor less than x. If not, then revert to the previous matrix. The choice of n^2 iterations is somewhat arbitrary, but it gives n attempts per column on average and requires relatively little preprocessing time. In addition, we also try this sharpening on the nearly doubly stochastic matrix, and pick the one with a tighter bound. We will refer to schemes that use this sharpening method by adding "-S" to their name. Note that while sharpening cannot worsen the basic, depth-0 upper bound, it is a priori not obvious whether this holds true also when taking deep bounds.

Empirical Results

We empirically compared the proposed upper bound and preprocessing method against previous state of the art. We also compared rejection sampling against importance sampling. Our implementations as well as the details of the computing infrastructure are included in the supplement.

Comparison of Rejection Samplers

We experimented with three rejection sampling schemes:

- ADAPART: An adaptive partitioning scheme based on the Brouwer–Schrijver bound.
- EXTHUBER: A scheme using the extended Huber bound.
- HL: A scheme based on the Huber–Law bound.

Each of the schemes uses the depth-20 variant of the corresponding bound. We also consider modifications of the schemes that preprocess the matrix using either of the described methods. Implementations of ADAPART and HL are by Harviainen et al. (2021). We extended their source code with EXTHUBER to enable deep bounds for it.

We experimented with three classes of matrices: Matrices in *Random Permutation* were generated such that we start with a matrix of zeros, choose $1 + \lceil \log_2 n \rceil$ permutations of $\{1, 2, \ldots, n\}$ uniformly at random, and then replace the corresponding entries in the matrix by $\mathrm{Uniform}[0, 1]$ distributed values for each permutation. *Block Diagonal* consists of $n \times n$ matrices with blocks of 5×5 matrices of

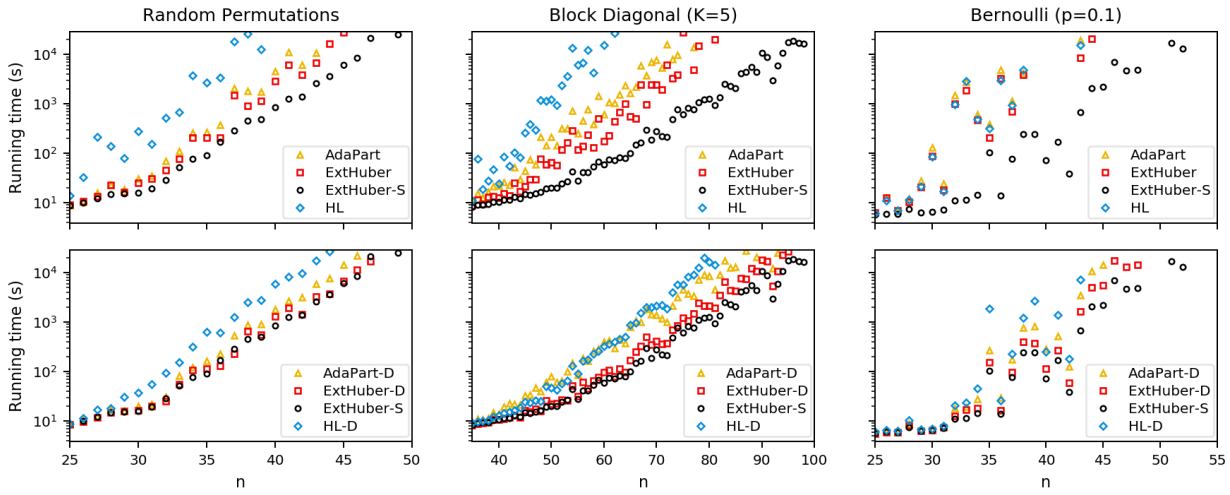

Figure 1: Estimates of the running times for three classes of matrices with (first row) and without (second row) preprocessing.

Uniform$[0, 1]$ distributed entries on the main diagonal and the rest are zeros. Finally, *Bernoulli* contains matrices whose each entry is 1 with probability 0.1 and 0 otherwise.

We followed the test setup of Harviainen et al. (2021) and evaluated our implementations by estimating the running time required for getting a $(0.1, 0.05)$-approximation, corresponding to 388 accepted draws with the GBAS algorithm. The estimate of the time was computed based on the time required for 65 accepted draws to save computation time. The running time was limited to 4825 seconds per instance, which would be about 8 hours when scaled for 388 draws. Figure 1 shows the running time estimates. The fluctuation in *Bernoulli* is due to high variance on how much each entry contributes to the value of the permanent. ExtHuber beats both other samplers on all instances, as suggested by our theoretical results, and at best the improvements are of one order of magnitude. While sharpening is helpful in nearly all instances, taking deep bounds diminishes its effects—with the basic depth-0 bounds, we observed sharpening to yield improvements by up to a factor of 20. On the other hand, the speedup often increases as n increases, suggesting further potential for improvements on larger matrices.

In addition, we tested the schemes on real-world matrices from Network Data Repository (Rossi and Ahmed 2015) licensed under CC-BY-SA. We included matrices, for which the permanent can be approximated in reasonable time, whereas Ryser's exact algorithm would be infeasible. Using publicly available instances for evaluating the performance of exact and approximation algorithms for permanents has been a common practice in recent works (Harviainen et al. 2021; Kuck et al. 2019a; Chakraborty et al. 2019). Our results are reported in Table 2. The running time was limited to 36 000 seconds per instance—computations exceeding this are marked with "-". Again, ExtHuber achieves improvements of an order of magnitude. In CAG-mat72, sharpening leads to a significant boost in the results as well.

Comparison Against Importance Sampling

Importance sampling schemes for approximating the permanent generate some number of independent permutations $\sigma_1, \sigma_2, \ldots, \sigma_m$ from an appropriate proposal distribution with a probability mass function q, and then estimate the permanent by the reweighted sample average $m^{-1} \sum_{j=1}^{m} w(\sigma_j)/q(\sigma_j)$.

Similar to rejection samplers, sequential importance samplers build the permutations iteratively by fixing one or more entries and then drawing a sample from the remaining submatrix. Most importance samplers in the literature appear to be designed for $(0, 1)$-matrices, but there are also some that have been developed for nonnegative matrices or that can be easily adapted for them. Here, we will compare ExtHuber-S against the following importance samplers:

SIS: A sequential importance sampler of Alimohammadi et al. (2021) that takes advantage of Sinkhorn balancing and always draws a permutation of positive weight.

PPS: A sequential importance sampler of Smith and Dawkins (2001) that randomly orders rows based on probabilities proportional to sums of their entries.

For SIS we modified the available source code slightly to have it take the weights of the permutations into account. For PPS we use our own implementation.

We analyzed the evolution of the estimates of the schemes over time. We show selected results in Figure 2. In addition to having an instance from *Random Permutations*, we experimented with three kinds of matrices we observed to be empirically hard: The first of them has many submatrices with zero permanent, the second one has a single permutation of high weight and many low-weight permutations, and the last one is a mix of two hard matrices. More precisely, the instance of *Random Entries* has $1 + \lfloor \log_2 n \rfloor$ Uniform$[0.5, 1.0]$ distributed entries on each row at positions that are sampled uniformly at random with replacement and the rest are zeros. The instance matrix A of *Random Lines* is a symmetric 50×50 matrix with Uniform$[0.5, 1.0]$ distributed entries on the

Instance	n	ExtHuber	ExtHuber-D	ExtHuber-S	AdaPart	AdaPart-D	HL	HL-D
ENZYMES-g283	52	-	$1 \cdot 10^2$	$\mathbf{5 \cdot 10^1}$	-	$3 \cdot 10^2$	-	$6 \cdot 10^2$
ENZYMES-g501	66	-	$3 \cdot 10^1$	$\mathbf{2 \cdot 10^1}$	-	$5 \cdot 10^1$	-	$5 \cdot 10^1$
ENZYMES-g575	51	$5 \cdot 10^3$	$1 \cdot 10^4$	$\mathbf{3 \cdot 10^3}$	$1 \cdot 10^4$	$3 \cdot 10^4$	$2 \cdot 10^4$	-
CAG-mat72	72	-	$1 \cdot 10^4$	$\mathbf{2 \cdot 10^3}$	-	$3 \cdot 10^4$	-	-
GD95-c	62	$6 \cdot 10^3$	-	$\mathbf{4 \cdot 10^3}$	$2 \cdot 10^4$	-	$2 \cdot 10^4$	-

Table 2: Running times of the rejection samplers for getting $(0.1, 0.05)$-approximations of permanents of real-world instances.

main diagonal. In addition, 16 integers x_i, y_i, and z_i are generated uniformly at random, and then the entries a_{x_i+j,y_i+j} with $j \in \{0, 1, \ldots, z_i\}$ are replaced by Uniform$[0.00, 0.25]$ distributed values. We also let a_{y_i+j,x_i+j} be a_{x_i+j,y_i+j} for each j. Again, the rest of the matrix is zeros. Finally, *Mixed* is a 50×50 matrix that starts with a 40×40 *Random Lines* instance, and its bottom right 10×10 submatrix has 2 entries per row sampled with replacement and then assigned a value between 0.5 and 1.0 uniformly at random.

On *Random Permutations*, ExtHuber-S is less stable than the importance samplers. On *Random Entries*, SIS beats others clearly as ExtHuber-S and PPS converge rather slowly. This may be due to the dead-ends for both self-reducible and sequential samplers. On *Random Lines*, the rejection sampler is competitive against the importance samplers, and samples from SIS have higher variance than those from PPS. Finally, on *Mixed* ExtHuber-S appears much stabler than importance samplers. To confirm this is not a fluke, we include additional evaluations for *Mixed* in supplemental material. In our preliminary experiments SIS tends to perform well in general; the purpose of our specific instances is to show there exist hard instances for it, too.

Concluding Remarks

We presented a new randomized approximation scheme for the permanent of nonnegative matrices. Our empirical results suggest that on hard instances our scheme is an order-of-magnitude faster than the previous best schemes. This improvement is explained by our new nesting upper bound that we dubbed Extended Huber: this bound is, on the one hand, faster to evaluate than the tighter (but generally non-nesting) Brouwer–Schrijver bound employed in AdaPart (Kuck et al. 2019a), and on the other hand, tighter than the equally fast Huber–Law bound (Huber and Law 2008). As our bound preserves the row-product structure, it can be boosted further, like the other mentioned bounds, using the recent "deep rejection sampling" technique (Harviainen et al. 2021). The shown improvements are also partly due to our efficient, heuristic implementation of sharpening, a preprocessing method for enhancing the upper bound.

We also compared our approximation scheme against two state-of-the-art importance sampling (IS) based estimators (Smith and Dawkins 2001; Alimohammadi et al. 2021). While these estimators do not enjoy accuracy guarantees, they performed relatively well in our selected tests, converging fast near the true permanent. However, we also exhibited matrices, for which one or the other estimator, or both, were inferior to our rejection sampling scheme, which *does*

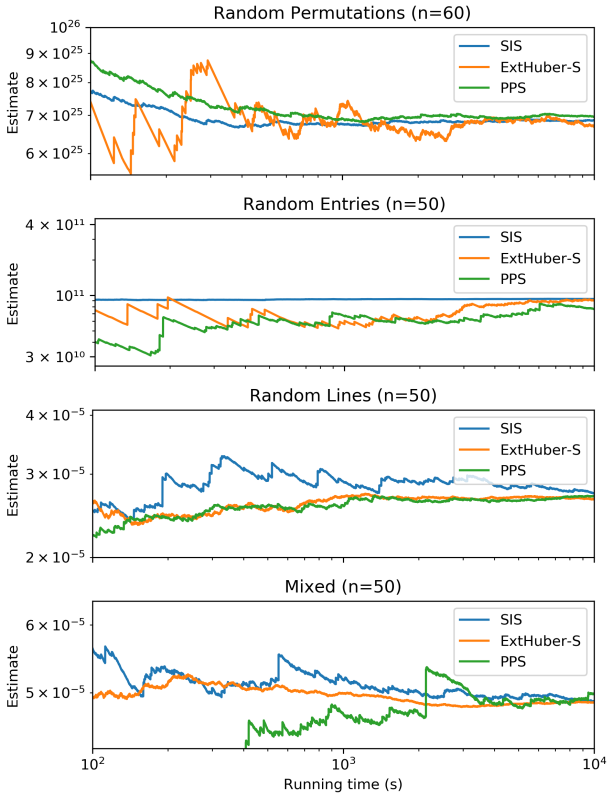

Figure 2: Evolution of the estimates for four matrices.

provide accuracy guarantees. This observation highlights the value of guarantees and suggests that it may be difficult, or even impossible, to equip IS based estimators with good sample complexity bounds or stopping rules.

The permanent is a special case of weighted model counting, a problem extensively studied in the AI research community. While our technical constructions are specific to the matrix permanent, we believe they may inspire similar constructions also for other concrete model counting problems. In particular, the technique of extending a "counting bound" to a "weighted counting bound" could be useful also in other contexts where good bounds are easier to discover for the unweighted, purely combinatorial objects—can we, for example, develop such bounds for independent sets (Dyer et al. 2021) or DAGs (Talvitie, Vuoksenmaa, and Koivisto 2019), and then generalize them to the weighted case?

Acknowledgments

Research partially supported by the Academy of Finland, Grant 316771.

References

Aaronson, S.; and Arkhipov, A. 2011. The computational complexity of linear optics. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, 333–342. ACM.

Alimohammadi, Y.; Diaconis, P.; Roghani, M.; and Saberi, A. 2021. Sequential importance sampling for estimating expectations over the space of perfect matchings. *CoRR*, abs/2107.00850.

Bezáková, I.; Štefankovič, D.; Vazirani, V. V.; and Vigoda, E. 2008. Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems. *SIAM J. Comput.*, 37(5): 1429–1454.

Bianco, G. L.; Lorca, X.; Truchet, C.; and Pesant, G. 2019. Revisiting Counting Solutions for the Global Cardinality Constraint. *J. Artif. Intell. Res.*, 66: 411–441.

Brègman, L. M. 1973. Some properties of nonnegative matrices and their permanents. *Dokl. Akad. Nauk*, 211(1): 27–30.

Chakraborty, S.; Fried, D.; Meel, K. S.; and Vardi, M. Y. 2015. From Weighted to Unweighted Model Counting. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 689–695. AAAI Press.

Chakraborty, S.; Shrotri, A. A.; and Vardi, M. Y. 2019. On Symbolic Approaches for Computing the Matrix Permanent. In Schiex, T.; and de Givry, S., eds., *Proceedings of the Twenty-Fifth International Conference on Principles and Practice of Constraint Programming*, volume 11802 of *Lecture Notes in Computer Science*, 71–90. Springer.

Chatterjee, S.; and Diaconis, P. 2018. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2): 1099 – 1135.

Dagum, P.; Karp, R. M.; Luby, M.; and Ross, S. M. 2000. An Optimal Algorithm for Monte Carlo Estimation. *SIAM J. Comput.*, 29(5): 1484–1496.

Dyer, M. E.; Jerrum, M.; Müller, H.; and Vuskovic, K. 2021. Counting Weighted Independent Sets beyond the Permanent. *SIAM J. Discret. Math.*, 35(2): 1503–1524.

Gomes, C. P.; Sabharwal, A.; and Selman, B. 2006. Model Counting: A New Strategy for Obtaining Good Bounds. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 54–61.

Harviainen, J.; Röyskö, A.; and Koivisto, M. 2021. Approximating the Permanent with Deep Rejection Sampling. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*.

Huber, M. 2006. Exact Sampling from Perfect Matchings of Dense Regular Bipartite Graphs. *Algorithmica*, 44(3): 183–193.

Huber, M. 2017. A Bernoulli mean estimate with known relative error distribution. *Random Struct. Algorithms*, 50(2): 173–182.

Huber, M.; and Law, J. 2008. Fast Approximation of the Permanent for Very Dense Problems. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA)*, 681–689. SIAM.

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

Kuck, J.; Dao, T.; Rezatofighi, H.; Sabharwal, A.; and Ermon, S. 2019a. Approximating the Permanent by Sampling from Adaptive Partitions. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 8860–8871. Curran Associates, Inc.

Kuck, J.; Dao, T.; Zhao, S.; Bartan, B.; Sabharwal, A.; and Ermon, S. 2019b. Adaptive Hashing for Model Counting. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Minc, H. 1963. Upper bounds for permanents of (0, 1)-matrices. *Bull. Amer. Math. Soc*, 69(6): 789–791.

Minc, H. 1984. *Permanents*. Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Newman, J. E.; and Vardi, M. Y. 2020. FPRAS Approximation of the Matrix Permanent in Practice. *CoRR*, abs/2012.03367.

Nijenhuis, A.; and Wilf, H. S. 1978. *Combinatorial algorithms: for computers and calculators*. Academic Press.

Rossi, R. A.; and Ahmed, N. K. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 4292–4293. AAAI Press.

Ryser, H. J. 1963. *Combinatorial mathematics*, volume 14. Mathematical Association of America.

Schrijver, A. 1978. A short proof of Minc's conjecture. *J. Comb. Theory Ser. A*, 25(1): 80–83.

Sinkhorn, R. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Stat.*, 35(2): 876–879.

Smith, P.; and Dawkins, B. 2001. Estimating the permanent by importance sampling from a finite population. *Journal of Statistical Computation and Simulation*, 70(3): 197–214.

Soos, M.; and Meel, K. S. 2019. BIRD: Engineering an Efficient CNF-XOR SAT Solver and its Applications to Approximate Model Counting. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, 1592–1599.

Soules, G. W. 1991. The rate of convergence of Sinkhorn balancing. *Linear Algebra Appl.*, 150: 3–40.

Soules, G. W. 2005. Permanental bounds for nonnegative matrices via decomposition. *Linear Algebra Appl.*, 393: 73–89.

Sullivan, F.; and Beichl, I. 2014. Permanents, α-permanents and Sinkhorn balancing. *Comput. Stat.*, 29(6): 1793–1798.

Talvitie, T.; Vuoksenmaa, A.; and Koivisto, M. 2019. Exact Sampling of Directed Acyclic Graphs from Modular Distributions. In Globerson, A.; and Silva, R., eds., *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 115 of *Proceedings of Machine Learning Research*, 965–974. AUAI Press.

Tassa, T. 2012. Finding all maximally-matchable edges in a bipartite graph. *Theor. Comput. Sci.*, 423: 50–58.

Uhlmann, J. K. 2004. Matrix permanent inequalities for approximating joint assignment matrices in tracking systems. *J. Frankl. Inst.*, 341(7): 569–593.

Valiant, L. G. 1979. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8(2): 189–201.