The Classical Complexity of Boson Sampling

Peter Clifford
Department of Statistics
University of Oxford
United Kingdom

Raphaël Clifford
Department of Computer Science
University of Bristol
United Kingdom

Abstract

We study the classical complexity of the exact Boson Sampling problem where the objective is to produce provably correct random samples from a particular quantum mechanical distribution. The computational framework was proposed by Aaronson and Arkhipov in 2011 as an attainable demonstration of 'quantum supremacy', that is a practical quantum computing experiment able to produce output at a speed beyond the reach of classical (that is non-quantum) computer hardware. Since its introduction Boson Sampling has been the subject of intense international research in the world of quantum computing. On the face of it, the problem is challenging for classical computation. Aaronson and Arkhipov show that exact Boson Sampling is not efficiently solvable by a classical computer unless $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ and the polynomial hierarchy collapses to the third level.

The fastest known exact classical algorithm for the standard Boson Sampling problem takes $\mathcal{O}(\binom{m+n-1}{n} n2^n)$ time to produce samples for a system with input size n and m output modes, making it infeasible for anything but the smallest values of n and m. We give an algorithm that is much faster, running in $\mathcal{O}(n2^n + \mathsf{poly}(m, n))$ time and $\mathcal{O}(m)$ additional space. The algorithm is simple to implement and has low constant factor overheads. As a consequence our classical algorithm is able to solve the exact Boson Sampling problem for system sizes far beyond current photonic quantum computing experimentation, thereby significantly reducing the likelihood of achieving near-term quantum supremacy in the context of Boson Sampling.

1 Introduction

The promise of exponentially faster quantum algorithms for problems of practical interest is one of the most exciting prospects in computer science. Most famously, a quantum computer would allow us to factorise integers in polynomial time (Shor, 1997) and perform unstructured search on an input of n elements in $\mathcal{O}(\sqrt{n})$ time (Grover, 1996). In the short term however, despite impressive progress in recent years, there are still considerable challenges to overcome before we can build a fully universal quantum computer. Until this time, the question of how to design quantum experiments which fall short of fully universal computation but which still show significant quantum speed-up over known classical (that is non-quantum) algorithms remains of great interest. In a breakthrough contribution in STOC '11 Aaronson and Arkhipov (2011) proposed an experimental set-up in linear optics known as Boson Sampling as a potentially practical way to do just this.

Since its introduction, Boson Sampling has attracted a great deal of attention with numerous experimental efforts around the world attempting implementations for various problem sizes (see e.g. Bentivegna et al., 2015; Spring et al., 2013; Broome et al., 2013; Tillmann et al., 2013; Crespi et al., 2013; Spagnolo et al., 2014; Latmiral et al., 2016). The ultimate goal is to exhibit a physical quantum experiment of such a scale that it would be hard if not impossible to simulate the output classically and thereby to establish so-called 'quantum supremacy'. In terms of the physical Boson Sampling experiment, n corresponds to the number of photons and m the number of output modes and increasing either of these is difficult in practice. Progress has therefore been slow (see Lund et al., 2017, and the references therein) with the current experimental record being $n = 5, m = 9$ (Wang et al., 2016).

Translated into conventional computing terms, the task is to generate independent random samples from a specific probability distribution on multisets of size n with elements in the range 1 through m. For Boson Sampling, the probability of each multiset is related to the permanent of an $n \times n$ matrix built from possibly repeated rows of a larger $m \times n$ matrix where the multiset determines the rows used in the construction. The *Boson Sampling problem* is to produce such samples either with a quantum photonic device or with classical computing hardware.

On the face of it, Boson Sampling is a challenging problem for classical computation. Aaronson and Arkhipov (2011) showed that exact Boson Sampling is not solvable in polynomial time by a classical computer unless $\mathsf{P}^{\#\mathsf{P}} = \mathsf{BPP}^{\mathsf{NP}}$ and the polynomial hierarchy collapses to the third level.

Without a clear understanding of the classical complexity of the problem it is difficult to determine the minimum experimental size needed to establish quantum supremacy. In the absence of a fast classical algorithm for Boson Sampling, speculation about the minimum size of n (with $m = n^2$) has varied a great deal in the literature from e.g. 7 (Latmiral et al., 2016), between 20 and 30 (Aaronson and Arkhipov, 2014) and all the way up to 50 (Lund et al., 2017). Broadly speaking the lowest estimate corresponds to the limits of the previous fastest classical Boson Sampling algorithm and the highest with an assumption that the polynomial hierarchy will not collapse.

Brute force evaluation of the probabilities of each multiset as a preliminary to random sampling, requires the calculation of $\binom{m+n-1}{n}$ permanents of $n \times n$ matrices, each one of which takes $\mathcal{O}(n2^n)$ time with the fastest known algorithms. If $m \geqslant n^2$, as considered by Aaronson and Arkhipov (2011), the total running time is $\Theta(2^n \mathrm{e}^n (m/n)^n n^{1/2})$. The space usage of such a naive approach can be reduced to the storage needed for the required sample size (see e.g. Efraimidis, 2015), however the running time for anything but the smallest values of n and m remains prohibitive. Previous to the work we present here, no faster provably correct classical Boson Sampling algorithm was known.

Recently Neville et al. (2017) gave strong numerical evidence that a specially designed MCMC implementation can sample approximately from the collision-free version of Boson Sampling. The time cost per sample for their MCMC method for problems with size $n \leqslant 30$ corresponds to that of computing around two hundred $n \times n$ permanents, making it computationally feasible for the problem sizes they consider. This was the first practical evidence that classical Boson Sampling might be possible faster than the naive brute force approach and is a significant motivation for the work we present here.

We show that exact Boson Sampling on a classical computer can indeed be performed dramatically faster than the naive algorithm, significantly raising the bar for the minimum size of quantum experiment needed to establish quantum supremacy. Our sampling algorithm also provides the probability associated with any given sample at no extra computational cost.

Theorem 1. *The time complexity of Boson Sampling is:*

$$\mathcal{O}(n2^n + \mathsf{poly}(m, n)),$$

where $\mathsf{poly}(m, n) = \mathcal{O}(mn^2)$. *The additional space complexity on top of that needed to store the input is* $\mathcal{O}(m)$.

A particularly attractive property of the running time we give is that the exponentially growing term depends only on n, the size of the multisets we wish to sample, and not m which is potentially much larger. In the most natural parameter regime when m is polynomial in n our algorithm is therefore approximately $\binom{m+n-1}{n}$ times faster than the fastest previous method. Our algorithm is both straightforward to implement and has small constant factor overheads, with the total computational cost to take one sample approximately equivalent to that of calculating two $n \times n$ permanents. This gives a remarkable resolution to the previously open question of the relationship between the classical complexity of computing the permanent of an $n \times n$ matrix and Boson Sampling.

Based on the time complexity we demonstrate in Theorem 1 and bearing in mind recent feasibility studies for the calculation of large permanents (Wu et al., 2016) we find support for $n = 50$ as the threshold for quantum supremacy for the exact Boson Sampling problem.

Our sampling algorithm also applies directly to the closely related problem of scattershot Boson Sampling (Bentivegna et al., 2015). From a mathematical perspective this produces no additional complications beyond the specification of a different $m \times n$ matrix. Once the matrix is specified we can apply our new sampling algorithm directly.

The algorithmic speed-up we achieve relies on a number of key innovations. As a first step we expand the sample space of multisets into a product set $[m]^n$, i.e. the space of all arrays $\mathbf{r} = (r_1, \ldots, r_n)$, with each element in the range 1 to m. This simple transformation allows us to expose combinatorial properties of the sampling problem which were otherwise opaque. We then develop explicit expressions for the probabilities of subsequences of arrays in this set. Application of the chain rule for conditional probabilities then leads to our first speed-up with an $\mathcal{O}(mn3^n)$ time algorithm for Boson Sampling. For our main result in Theorem 1, we introduce two further innovations. First we exploit the Laplace expansion of a permanent to give an efficient amortised algorithm for the computation of the permanent of a number of closely related matrices. We then expand the sample space further with an auxiliary array representing column indices and show how a conditioning argument allows Boson Sampling to be viewed as sampling from a succession of coupled conditional distributions. The final result is our improved time complexity of $\mathcal{O}(n2^n + mn^2)$.

A consequence of our analysis is that we are also able to provide an efficiently computable bound on the probability that a sampled multiset drawn from the Boson Sampling distribution will contain duplicated elements. This enables us to bound the performance of rejection sampling variants of our sampling algorithm when applied to the so-called ‘collision-free’ setting described by Aaronson and Arkhipov.

One further application of our new sampling algorithm lies in tackling the question of whether the output of physical Boson Sampling devices correspond to theoretical predictions. There has been considerable interest in this area (Aaronson and Arkhipov, 2014; Tichy et al., 2014; Wang and Duan, 2016; Walschaers et al., 2016) however previously developed methods have had to work under the assumption that we cannot sample from the Boson Sampling distribution for even moderate sizes of n and m. Armed with our new sampling algorithm, we can now sample directly from the Boson Sampling distribution for a much larger range of parameters, allowing a far greater range of statistical tests to be applied.

Related work and computation of the permanent

There is an extensive literature on the problems of sampling from unwieldy distributions with applications in all the main scientific disciplines. Much of this work has relied on Markov chain Monte Carlo (MCMC) methods where the target distribution can be identified as the equilibrium distribution of a Markov chain (see e.g. Hastings, 1970). For Boson Sampling the most closely related work to ours is that of Neville et al. (2017) mentioned earlier. In their paper the authors also set out the hurdles that need to be overcome for a physical Boson Sampling experiment to reach higher values of n. Their MCMC approach does not however permit them to estimate how well their sampling method approximates the true Boson Sampling distribution, other than through numerical experimentation. Our new exact sampling algorithm is therefore the first provably correct algorithmic speed-up for Boson Sampling. In fact our algorithm also has smaller constant factor overheads, costing approximately two permanent calculations of $n \times n$ matrices for all values of n (assuming m remains polynomial in n).

There are MCMC algorithms which permit exact sampling, for example via ‘coupling from the past’, where guarantees of performance can be provided (Propp and Wilson, 1996). However, the range of problems for which this approach has proven applicable is necessarily more limited and there has yet to be a successful demonstration for the Boson Sampling problem.

Permanents: Computation of the permanent of a matrix was shown to be #P-hard by Valiant (1979) and hence is unlikely to have a polynomial time solution. Previously Ryser (1963) had proposed an $O(n^2 2^n)$ time algorithm to compute the permanent of an $n \times n$ matrix. This was sped up to $O(n 2^n)$ time by Nijenhuis and Wilf (1978) and many years later a related formula with the same time complexity was given by Glynn (2010). It is Glynn's computational formula for the permanent which we will use as the basis of our sampling algorithms.

Let $M = (m_{i,j})$ be an $n \times n$ matrix with $m_{i,j} \in \mathbb{C}$, and let Per M denote its permanent, then

$$\operatorname{Per} M = \frac{1}{2^{n-1}} \sum_{\delta} \left(\prod_{k=1}^{n} \delta_k \right) \prod_{j=1}^{n} \sum_{i=1}^{n} \delta_i m_{i,j} \tag{1}$$

where $\delta \in \{-1, 1\}^n$ with $\delta_1 = 1$.

A naive implementation of this formula would require $\mathcal{O}(n^2 2^n)$ time however by considering the δ arrays in Gray code order this is reduced to $O(n 2^n)$ time. Note that further reduction in the time complexity may be possible when M has repeated rows or columns; see Tichy (2011, Appendix B) and Shchesnovich (2013, Appendix D). In the special case of computation of the permanent over $\mathsf{poly}(n)$ bit integers, Björklund (2016) has a faster $\mathcal{O}(n 2^{n - \Omega(\sqrt{n / \log n})})$ time algorithm for computing the permanent.

2 Problem specification

In mathematical terms the Boson Sampling problem is as follows.

Let m and n be positive integers and consider all multisets of size n with elements in $[m]$, where $[m] = \{1, \dots, m\}$. Each multiset can be represented by an array $\mathbf{z} = (z_1, \dots, z_n)$ consisting of elements of the multiset in non-decreasing order: $z_1 \leqslant z_2 \leqslant \dots \leqslant z_n$. We denote the set of distinct values of $\mathbf{z}$ by $\Phi_{m,n}$. The cardinality of $\Phi_{m,n}$ is known to be $\binom{m+n-1}{n}$ – see Feller (1968), for example. We define $\mu(\mathbf{z}) = \prod_{j=1}^{m} s_j!$ where s_j is the multiplicity of the value j in $\mathbf{z}$.

Now let $A = (a_{ij})$ be a complex valued $m \times n$ matrix consisting of the first n columns of a given $m \times m$ Haar random unitary matrix. For each $\mathbf{z}$, build an $n \times n$ matrix $A_{\mathbf{z}}$ where the k-th row of $A_{\mathbf{z}}$ is row z_k in A for $k = 1, \dots, n$ and define a probability mass function (pmf) on $\Phi_{m,n}$ as

$$q(\mathbf{z}) = \frac{1}{\mu(\mathbf{z})} |\operatorname{Per} A_{\mathbf{z}}|^2 \overset{\text{defn}}{=} \frac{1}{\mu(\mathbf{z})} \left| \sum_{\sigma} \prod_{k=1}^{n} a_{z_k \sigma_k} \right|^2, \quad \mathbf{z} \in \Phi_{m,n}, \tag{2}$$

where Per $A_{\mathbf{z}}$ is the permanent of $A_{\mathbf{z}}$ and in the definition the summation is for all $\sigma \in \pi[n]$, the set of permutations of $[n]$.

The Cauchy-Binet formula for permanents (see Marcus and Minc, 1965, page 579) can be used to show the function q is indeed a pmf. There is a demonstration from first principles in the proof of Lemma 1. See also Aaronson and Arkhipov (2011, Theorem 3.10) for a demonstration from physical principles.

The computing task is to simulate random samples from the pmf $q(\mathbf{z})$.

3 Exact Boson Sampling Algorithms

We give two algorithms for Boson Sampling[1]. Algorithm A runs in $\mathcal{O}(mn3^n)$ time to create a single sample. This is already a significant speed-up over the $\Theta\big(\binom{m+n-1}{n}n2^n\big)$ time complexity of the fastest solution in the literature. Algorithm A prepares the way for Algorithm B our main result with running time specified in Theorem 1.

We will repeatedly need to sample from an unnormalised discrete probability mass function (weight function). To establish the complexity of our Boson Sampling algorithms we need only employ the most naive linear time method although faster methods do exist. In particular, if the probability mass function is described by an array of length m then it is possible to sample in $\mathcal{O}(1)$ time after $\mathcal{O}(m)$ preprocessing time, using Walker's alias method (Walker, 1974; Kronmal and Jr., 1979).

Expanding the sample space: We start by expanding the sample space $\Phi_{m,n}$ and consider a distribution on the product space $[m]^n$, i.e. the space of all arrays $\mathbf{r} = (r_1, \dots, r_n)$, with each element in the range 1 to m. Since the permanent of a matrix is invariant to row reordering, we have $\operatorname{Per} A_{\mathbf{r}} = \operatorname{Per} A_{\mathbf{z}}$ when $\mathbf{r}$ is any permutation of $\mathbf{z}$. Furthermore the number of distinguishable rearrangements of $\mathbf{z}$ is $n!/\mu(\mathbf{z})$.

We claim that in order to sample from $q(\mathbf{z})$ we can equivalently sample from the pmf

$$p(\mathbf{r}) = \frac{1}{n!}\left|\operatorname{Per} A_{\mathbf{r}}\right|^2 = \frac{1}{n!}\left|\sum_{\sigma}\prod_{i=1}^{n} a_{r_i\sigma_i}\right|^2, \quad \mathbf{r} \in [m]^n. \tag{3}$$

In other words sample $\mathbf{r}$ from $p(\mathbf{r})$ and then rearrange the elements of $\mathbf{r}$ in non-decreasing order to give $\mathbf{z}$.

This follows since for any $\mathbf{z}$ there are $n!/\mu(\mathbf{z})$ equally likely values of $\mathbf{r}$ in the expanded sample space, i.e. $p(\mathbf{r}) = p(\mathbf{z})$ for all such values of $\mathbf{r}$. So by the addition rule for probabilities

$$q(\mathbf{z}) = \frac{n!}{\mu(\mathbf{z})}p(\mathbf{z}) = \frac{n!}{\mu(\mathbf{z})}\frac{1}{n!}\left|\operatorname{Per} A_{\mathbf{z}}\right|^2 = \frac{1}{\mu(\mathbf{z})}\left|\operatorname{Per} A_{\mathbf{z}}\right|^2, \quad \mathbf{z} \in \Phi_{m,n},$$

as claimed.

This straightforward reformulation will make the task of efficient Boson Sampling significantly simpler. Our approach focuses on the pmf $p(\mathbf{r})$.

We begin by deriving the marginal pmfs of the leading subsequences of $(r_1, \dots, r_n)$ in the following lemma. This will then provide the pmfs of successive elements of $(r_1, \dots, r_n)$ conditional on previous values. In that way we are able to rewrite $p(r_1, \dots, r_n)$ as a product of conditional pmfs, using the chain rule of probability, and exploit the chain for progressively simulating $r_1, \dots, r_n$.

Lemma 1. *The joint pmf of the subsequence $(r_1, \dots, r_k)$ is given by*

$$p(r_1, \dots, r_k) = \frac{(n-k)!}{n!}\sum_{c\in\mathcal{C}_k}\left|\operatorname{Per} A^c_{r_1,\dots,r_k}\right|^2, \quad k = 1, \dots, n,$$

where $\mathcal{C}_k$ is the set of k-combinations taken without replacement from $[n]$ and $A^c_{r_1,\dots,r_k}$ is the matrix formed from rows $(r_1, \dots, r_k)$ of the columns c of A.

Proof. The convention will be that c is a set and $(r_1, \dots, r_k)$ is an array with possibly multiple instances of its elements. The invariance of permanents under column and row permutations removes any ambiguity in the interpretation of $\operatorname{Per} A^c_{r_1,\dots,r_k}$.

[1] Implementations of these algorithms are available from the authors on request

For completeness we start by confirming that $p(\mathbf{r})$ is a pmf, in other words $\sum_{\mathbf{r}} p(\mathbf{r}) = 1$. From the definition of $p(\mathbf{r})$ and multiplying by $n!$, we have

$$\begin{aligned} n!\sum_{\mathbf{r}} p(\mathbf{r}) &= \sum_{\mathbf{r}} \left| \sum_{\sigma} \prod_{i=1}^{n} a_{r_i\sigma_i} \right|^2 , \sigma \in \pi[n] \\ &= \sum_{\mathbf{r}} \left(\sum_{\sigma} \prod_{i=1}^{n} a_{r_i\sigma_i} \right) \left(\sum_{\tau} \prod_{i=1}^{n} \bar{a}_{r_i\tau_i} \right) , \tau \in \pi[n] \\ &= \sum_{\sigma} \sum_{\tau} \left(\sum_{\mathbf{r}} \prod_{i=1}^{n} a_{r_i\sigma_i} \bar{a}_{r_i\tau_i} \right) \\ &= \sum_{\sigma} \sum_{\tau} \prod_{i=1}^{n} \sum_{k=1}^{m} a_{k,\sigma_i} \bar{a}_{k\tau_i} . \end{aligned}$$

The product is 0 when $\sigma_i \neq \tau_i$ for any i, by the orthonormal property of A. Otherwise when $\sigma = \tau$ the product is 1 so the final expression reduces to $n!$ and $\sum_{\mathbf{r}} p(\mathbf{r}) = 1$ as expected.

Now looking at the case $k = 1$ we have similarly

$$\begin{aligned} n!p(r_1) &= \sum_{r_2 \ldots r_n} \left(\sum_{\sigma} \prod_{i=1}^{n} a_{r_i\sigma_i} \right) \left(\sum_{\tau} \prod_{i=1}^{n} \bar{a}_{r_i\tau_i} \right) \\ &= \sum_{\sigma} \sum_{\tau} a_{r_1\sigma_1} \bar{a}_{r_1\tau_1} \prod_{i=2}^{n} \sum_{k=1}^{m} a_{k,\sigma_i} \bar{a}_{k\tau_i} . \end{aligned}$$

Again for this to be non-zero, each of the terms in the product must be 1. This means $\sigma_i = \tau_i, i = 2, \ldots, n$ which implies $\sigma = \tau$.

It follows that

$$n!p(r_1) = \sum_{\sigma} |a_{r_1\sigma_1}|^2 = (n-1)! \sum_{k=1}^{n} |a_{r_1,k}|^2 ,$$

as claimed, so that

$$p(r_1) = \frac{1}{n} \left[|a_{r_1,1}|^2 + \ldots , + |a_{r_1,n}|^2 \right] , \; r_1 \in [m],$$

the sum of the squared moduli of elements of row r_1 divided by n.

In the same way,

$$n!p(r_1, r_2) = \sum_{\sigma} \sum_{\tau} a_{r_1\sigma_1} \bar{a}_{r_1\tau_1} a_{r_2\sigma_2} \bar{a}_{r_2\tau_2} \left(\prod_{i=3}^{n} \sum_{k=1}^{m} a_{k,\sigma_i} \bar{a}_{k\tau_i} \right) .$$

Again the only non-zero case is when $\sigma_i = \tau_i, i = 3, \ldots, n$. However it does not imply that $\sigma = \tau$, since we can also have $\sigma_1 = \tau_2, \sigma_2 = \tau_1$.

Consequently,

$$\begin{aligned} n!p(r_1, r_2) &= \sum_{\substack{\sigma \quad \tau \\ \sigma_3, \ldots, \sigma_n = \tau_3, \ldots, \tau_n}} \!\! a_{r_1\sigma_1} \bar{a}_{r_1\tau_1} a_{r_2\sigma_2} \bar{a}_{r_2\tau_2} \\ &= (n-2)! \sum_{c \in \mathcal{C}_2} \left(\sum_{\sigma \in \pi(c)} a_{r_1\sigma_1} \bar{a}_{r_2\sigma_2} \right)^2 \\ &= (n-2)! \sum_{c \in \mathcal{C}_2} \left| \text{Per}\, A^c_{r_1, r_2} \right|^2 , \end{aligned}$$

where $\mathcal{C}_2$ is the set of 2-combinations (without replacement) from $[n]$, $\pi(c)$ is the set of permutations of c and $A^c_{r_1,r_2}$ is a matrix formed from rows (r_1, r_2) and columns c of A.

In the same way

$$n!p(r_1, \dots, r_k) = (n-k)! \sum_{c \in \mathcal{C}_k} \left| \operatorname{Per} A^c_{r_1,\dots,r_k} \right|^2,$$

so that

$$p(r_1, \dots, r_k) = \frac{(n-k)!}{n!} \sum_{c \in \mathcal{C}_k} \left| \operatorname{Per} A^c_{r_1,\dots,r_k} \right|^2, \quad k = 1, \dots, n.$$

□

Our first Boson Sampling algorithm which we term Algorithm A samples from $p(\mathbf{r})$ by expressing this as a chain of conditional pmfs,

$$p(\mathbf{r}) = p(r_1)p(r_2|r_1)p(r_3|r_1, r_2) \dots p(r_n|r_1, r_2, \dots, r_{n-1}).$$

Sample r_1 from $p(r_1)$, $r_1 \in [m]$. Subsequently for stages $k = 2, \dots, n$, sample r_k from the conditional pmf $p(r_k|r_1, \dots, r_{k-1})$, $r_k \in [m]$ with $(r_1, \dots, r_{k-1})$ fixed. At completion, the array $(r_1, \dots, r_n)$ at stage n will have been sampled from the pmf $p(\mathbf{r})$. Sorting $(r_1, \dots, r_n)$ in non-decreasing order, the resulting multiset $\mathbf{z}$ is sampled from the Boson Sampling distribution $q(\mathbf{z})$.

Algorithm A Boson Sampler: single sample $\mathbf{z}$ from q($\mathbf{z}$) in $\mathcal{O}(mn3^n)$ time

```
Require: $m$ and $n$ positive integers; $A$ first $n$ columns of $m \times m$ Haar random unitary matrix
$\mathbf{r} \leftarrow \varnothing$                                            ▷ EMPTY ARRAY
FOR $k \leftarrow 1$ TO $n$ DO
    $w_i \leftarrow \sum_{c \in \mathcal{C}_k} |\operatorname{Per} A^c_{(\mathbf{r},i)}|^2, i \in [m]$        ▷ MAKE INDEXED WEIGHT ARRAY $w$
    $x \leftarrow$ SAMPLE$(w)$                                 ▷ SAMPLE INDEX $x$ FROM $w$
    $\mathbf{r} \leftarrow (\mathbf{r}, x)$                                    ▷ APPEND $x$ TO $\mathbf{r}$
END FOR
$\mathbf{z} \leftarrow$ INCSORT$(\mathbf{r})$                      ▷ SORT $\mathbf{r}$ IN NON-DECREASING ORDER
RETURN $\mathbf{z}$
```

Theorem 2. *Algorithm A samples from the Boson Sampling distribution with time complexity $\mathcal{O}(mn3^n)$ per sample. The additional space complexity of Algorithm A on top of that needed to store the input is $\mathcal{O}(m)$ words.*

Proof. The correctness follows from the chain rule for conditional probabilities and so we now analyse the running time. From Lemma 1, evaluation of the pmf $p(r_1)$ for $r_1 \in [m]$ involves the sum of squared moduli in each row of A, a total of $\mathcal{O}(mn)$ operations. Sampling from this pmf takes $\mathcal{O}(m)$ time. More generally we can write $p(r_k|r_1, \dots, r_{k-1}) = p(r_1, \dots, r_k)/p(r_1, \dots, r_{k-1})$. Note that r_k does not appear in the denominator, so to sample r_k from this conditional pmf we can equivalently sample from the pmf that is proportional to the numerator, considered purely as function of r_k with $r_1, \dots, r_{k-1}$ fixed. The numerator involves the calculating of several $k \times k$ permanents. With the best known algorithms for computing the permanent, each requires $\mathcal{O}(k2^k)$ operations. There are $\binom{n}{k}$ terms in $\mathcal{C}_k$ for each value of k, making a combined operation count at stage k of $\mathcal{O}(m\binom{n}{k}k2^k)$. Using

$$\sum_{k=1}^{n} k2^k \binom{n}{k} = \frac{2}{3}n3^n,$$

the total operation count is then $\mathcal{O}(mn3^n)$. We only need to store the result of one permanent calculation at a time and also one list of m probabilities at a time. This gives a total space usage of $\mathcal{O}(m)$ excluding the space needed to store the original matrix A. □

We now prepare to prove Theorem 1.

Laplace expansion: We make extensive use of the Laplace expansion for permanents (see Marcus and Minc, 1965, page 578), namely that for any $k \times k$ matrix $B = (b_{i,j})$,

$$\operatorname{Per} B = \sum_{\ell=1}^{k} b_{k,\ell} \operatorname{Per} B^{\diamond}_{k,\ell},$$

where $B^{\diamond}_{k,\ell}$ is the submatrix with row k and column ℓ removed. Note that $B^{\diamond}_{k,\ell}$ only depends on $B^{\diamond}_{k}$ the submatrix of B with the k-th row removed. An important consequence is that when B is modified by changing its k-th row, the modified permanent can be calculated in $\mathcal{O}(k)$ steps, provided the values $\{\operatorname{Per} B^{\diamond}_{k,\ell}\}$ are available.

For a fixed k we show that when B is formed from rows and columns of a Haar random unitary matrix, the values $\{\operatorname{Per} B^{\diamond}_{k,\ell}\}$ can in fact be calculated for all $\ell \in [k]$ in the same time as it takes to compute the permanent of a single $k \times k$ matrix. We will later take advantage of this fast amortised algorithm in the proof of Theorem 1 to quickly compute a set of permanents of matrices each with one row differing from the other.

Lemma 2. *For a fixed k, let B be a $k \times k$ matrix formed by selecting k not necessarily distinct rows from k distinct columns of a Haar random unitary matrix A and let $\{B^{\diamond}_{k,\ell}\}$ be submatrices of B with row k and column ℓ removed, $\ell \in [k]$. The collection $\{\operatorname{Per} B^{\diamond}_{k,\ell}, \ell \in [k]\}$ can be evaluated jointly in $\mathcal{O}(k2^k)$ operations and $\mathcal{O}(k)$ additional space.*

Proof of Lemma. By applying Glynn's formula (1) for a given value of ℓ we have:

$$\operatorname{Per} B^{\diamond}_{k,\ell} = \frac{1}{2^{k-2}} \sum_{\delta} \left(\prod_{i=1}^{k-1} \delta_i \right) \prod_{j \in [k] \setminus \ell} v_j(\delta), \quad \ell \in [k],$$

where $\delta \in \{-1, 1\}^{k-1}$ with $\delta_1 = 1$ and $v_j(\delta) = \sum_{i=1}^{k-1} \delta_i b_{ij}$.

Exploiting the usual trick of working through values of δ in Gray code order, the terms $\{v_j(\delta), j \in [k]\}$ can be evaluated in $\mathcal{O}(k)$ combined time for every new δ. This is because successive δ arrays will differ in one element. The product of the $v_j(\delta)$ terms can therefore also be computed in $\mathcal{O}(k)$ time giving $\mathcal{O}(k2^k)$ time to compute $\operatorname{Per} B^{\diamond}_{k,\ell}$ for a single value of ℓ, but of course this has to be replicated k times to give all values of ℓ.

In order to compute $\operatorname{Per} B^{\diamond}_{k,\ell}$ more efficiently for all ℓ we observe that

$$\prod_{j \in [k] \setminus \ell} v_j(\delta) = \frac{1}{v_\ell(\delta)} \prod_{j \in [k]} v_j(\delta)$$

whenever $v_\ell(\delta) \neq 0$. In this case we can simply evaluate the product on the right-hand side of this equation once and divide by $v_\ell(\delta)$ for each value of ℓ in turn, maintaining an array of length k of partial sums, with one entry for each value of ℓ.

Suppose on the other hand $v_\ell(\delta) = 0$ for some value of ℓ and δ. Recalling that $B^{\diamond}_{k}$ is the submatrix of B with the last row removed, this means that some δ-weighted linear combination of the elements in a column of $B^{\diamond}_{k}$ is zero. When the rows of $B^{\diamond}_{k}$ are distinct this has probability zero under the Haar random unitary assumption; similarly when the multiplicity of any of the rows is odd. The remaining possibility is that all the rows have even multiplicity and that δ lies in the left null space of $B^{\diamond}_{k}$. In which case $\sum_{i=1}^{k-1} \delta_i b_{ij} = 0$ for all j and hence $\prod_{j \in [k] \setminus \ell} v_j(\delta) = 0$ for all ℓ.

We can therefore always compute $\{\prod_{j \in [k] \setminus \ell} v_j(\delta), \ell \in [k]\}$ in $\mathcal{O}(k)$ time, giving an overall total time complexity of $\mathcal{O}(k2^k)$ for computing $\{\operatorname{Per} B^{\diamond}_{k,\ell}, \ell \in [k]\}$. Other than the original matrix, space used is dominated by the array of k partial sums. □

Furthermore the computation time has the same constant factor overhead as that of computing $\operatorname{Per} B$ (see Appendix).

Auxiliary column indices: We now expand the sample space even further with an auxiliary array $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$, where $\boldsymbol{\alpha} \in \pi[n]$, the set of permutations of $[n]$.

As with Lemma 1, the purpose is to create a succession of pmfs for leading subsequences of the array $(r_1, \dots, r_n)$ and thereby provide successive conditional pmfs so that $r_1, \dots, r_n$ can be simulated progressively. The novelty here is to introduce an overall conditioning variable $\boldsymbol{\alpha}$ so that sampling can be carried out more rapidly under that condition while demonstrating that the marginal distribution still correctly provides the pmf $p(\mathbf{r})$ in (3).

Define

$$\phi(r_1, \dots, r_k|\boldsymbol{\alpha}) = \frac{1}{k!}\left|\operatorname{Per} A^{[n]\setminus\{\alpha_{k+1},\dots,\alpha_n\}}_{r_1,\dots,r_k}\right|^2, \quad k = 1, \dots, n-1.$$

For notational convenience let $e_k = \phi(r_1, \dots, r_k|\boldsymbol{\alpha})$ and $d_k = \sum_{r_k} \phi(r_1, \dots, r_k|\boldsymbol{\alpha})$ for $k = 1, \dots, n-1$ with $e_n = p(r_1, \dots, r_n)$ and $d_n = p(r_1, \dots, r_{n-1})$. Note that $\phi(r_1, \dots, r_k|\boldsymbol{\alpha})$ is a pmf on $[m]^k$, equivalent to (3) using the subset $\{\alpha_1, \dots, \alpha_k\}$ of columns of A, so that in particular $d_1 = 1$.

Lemma 3. *With the preceding notation, let $\phi(\mathbf{r}|\boldsymbol{\alpha}) = \prod_{k=1}^{n} e_k/d_k$ then $p(\mathbf{r}) = \mathbb{E}_{\boldsymbol{\alpha}}\{\phi(\mathbf{r}|\boldsymbol{\alpha})\}$ where the expectation is taken over $\boldsymbol{\alpha}$, uniformly distributed on $\pi[n]$ for fixed $\mathbf{r}$.*

Proof. For the particular case $k = n-1$, Lemma 1 shows that $p(r_1, \dots, r_{n-1})$ can be written as a mixture of component pmfs and expressed as an expectation over α_n, i.e.

$$p(r_1, \dots, r_{n-1}) = \frac{1}{n}\sum_{j=1}^{n} \frac{1}{(n-1)!}\left|\operatorname{Per} A^{[n]\setminus j}_{r_1,\dots,r_{n-1}}\right|^2 = \mathbb{E}_{\alpha_n}\{\phi(r_1, \dots, r_{n-1}|\boldsymbol{\alpha})\},$$

where α_n is uniformly distributed on $[n]$. In other words d_n is the expectation of e_{n-1} over α_n. More generally when the leading subsequence is $(r_1, \dots, r_k)$ and the active columns of A are $[n] \setminus \{\alpha_{k+1}, \dots, \alpha_n\}$, we have

$$\sum_{r_k} \phi(r_1, \dots, r_k|\boldsymbol{\alpha}) = \mathbb{E}_{\alpha_k|\alpha_{k+1},\dots,\alpha_n}\{\phi(r_1, \dots, r_{k-1}|\boldsymbol{\alpha})\}, \quad k = 2, \dots, n-1,$$

where the expectation is taken over α_k uniformly distributed on $[n] \setminus \{\alpha_{k+1}, \dots, \alpha_n\}$ conditional on $(\alpha_{k+1}, \dots, \alpha_n)$ for fixed $\mathbf{r}$. This means d_k is the conditional expectation of e_{k-1}.

We now rewrite $\phi(\mathbf{r}|\boldsymbol{\alpha})$ as $e_n \prod_{k=2}^{n} e_{k-1}/d_k$ since $d_1 = 1$ and start by considering the expectations of the terms $e_{k-1}/d_k, k = 2, \dots, n$ with respect to α_2 given $\alpha_3, \dots, \alpha_n$. With the exception of e_1 these terms are specified by subsets of $\{\alpha_3, \dots, \alpha_n\}$, so they remain fixed under the conditioning. Since the conditional expectation of e_1 is d_2, the conditional expectation of $e_n \prod_{k=2}^{n} e_{k-1}/d_k$ is $e_n \prod_{k=3}^{n} e_{k-1}/d_k$, i.e. there is one fewer term in the product. Proceeding in this way and successively taking conditional expectations with respect to α_{k-1} uniformly distributed on $[n] \setminus \{\alpha_k, \dots, \alpha_n\}$ conditional on $(\alpha_k, \dots, \alpha_n)$, the product telescopes to $e_n(e_{n-1}/d_n) = p(r_1, \dots, r_n)e_{n-1}/d_n$. Finally taking expectations over α_n, this reduces to $p(r_1, \dots, r_n)$, since d_n is the expected value of e_{n-1}. Note a random permutation $(\alpha_1, \dots, \alpha_n)$ is generated starting from α_n in reverse order by successively selecting elements for inclusion at random from what remains. By the chain rule of expectation, we have then shown that $\mathbb{E}_{\boldsymbol{\alpha}}\{\phi(\mathbf{r}|\boldsymbol{\alpha})\} = p(\mathbf{r})$ as claimed. □

We can now describe our second and faster Boson Sampling algorithm. This algorithm is the basis of our main result, Theorem 1. We start by sampling a random permutation $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ uniformly distributed on $\pi[n]$. As in Algorithm A we consider a chain of conditional pmfs,

$$\phi(\mathbf{r}|\boldsymbol{\alpha}) = \prod_{k=1}^{n} e_k/d_k = \phi(r_1|\boldsymbol{\alpha})\phi(r_2|r_1,\boldsymbol{\alpha})\phi(r_3|r_1,r_2,\boldsymbol{\alpha})\dots\phi(r_n|r_1,r_2,\dots,r_{n-1},\boldsymbol{\alpha}),$$

where now $\phi(r_k|r_1,\dots,r_k,\boldsymbol{\alpha}) = e_k/d_k$, with the notation defined prior to Lemma 3.

At stage 1, we sample r_1 from the pmf $\phi(r_1|\boldsymbol{\alpha})$. Subsequently for stages $k = 2,\dots,n$, we sample r_k from $\phi(r_k|r_1,\dots,r_{k-1},\boldsymbol{\alpha})$ with $(r_1,\dots,r_{k-1})$ fixed, exploiting the Laplace expansion for permanents. At completion, the array $(r_1,\dots,r_n)$ at stage n will have been sampled from the pmf $p(\mathbf{r})$. Sorting $(r_1,\dots,r_n)$ in non-decreasing order, the resulting multiset $\mathbf{z}$ is sampled from the Boson Sampling distribution $q(\mathbf{z})$.

Algorithm B Boson sampler: single sample $\mathbf{z}$ from q($\mathbf{z}$) in $\mathcal{O}(n2^n + \mathsf{poly}(m,n))$ time

```
Require: $m$ and $n$ positive integers; $A$ first $n$ columns of $m \times m$ Haar random unitary matrix
$\mathbf{r} \leftarrow \varnothing$                                          ▷ EMPTY ARRAY
$A \leftarrow$ PERMUTE($A$)                                  ▷ RANDOMLY PERMUTE COLUMNS OF $A$
$w_i \leftarrow |a_{i,1}|^2, i \in [m]$                           ▷ MAKE INDEXED WEIGHT ARRAY $w$
$x \leftarrow$ SAMPLE($w$)                                   ▷ SAMPLE INDEX $x$ FROM $w$
$\mathbf{r} \leftarrow (\mathbf{r}, x)$                                       ▷ APPEND $x$ TO $\mathbf{r}$
FOR $k \leftarrow 2$ TO $n$ DO
    $B^{\diamond}_k \leftarrow A^{[k]}_{\mathbf{r}}$
    COMPUTE $\{\mathrm{Per}\, B^{\diamond}_{k,\ell}, \ell \in [k]\}$                        ▷ AS LEMMA 2
    $w_i \leftarrow |\sum_{\ell=1}^{k} a_{i,\ell}\,\mathrm{Per}\, B^{\diamond}_{k,\ell}|^2,\ i \in [m]$           ▷ USING LAPLACE EXPANSION
    $x \leftarrow$ SAMPLE($w$)
    $\mathbf{r} \leftarrow (\mathbf{r}, x)$
END FOR
$\mathbf{z} \leftarrow$ INCSORT($\mathbf{r}$)                               ▷ SORT $\mathbf{r}$ IN NON-DECREASING ORDER
RETURN $\mathbf{z}$
```

Proof of Theorem 1. It follows from the chain rule for conditional probabilities that for a given fixed $\boldsymbol{\alpha}$ Algorithm B samples from $\phi(\mathbf{r}|\boldsymbol{\alpha})$. As $\boldsymbol{\alpha}$ is uniformly distributed on $\pi[n]$ the algorithm therefore samples from the marginal distribution $\mathbb{E}_{\boldsymbol{\alpha}}\{\phi(\mathbf{r}|\boldsymbol{\alpha})\}$ which by Lemma 3 is the pmf $p(\mathbf{r})$ in (3). The complexity is established as follows. At stage 1, the pmf to be sampled is

$$\phi(r_1|\boldsymbol{\alpha}) = |\,\mathrm{Per}\, A^{[n]\setminus\{\alpha_2,\dots,\alpha_n\}}_{r_1}|^2 = |\,\mathrm{Per}\, A^{\alpha_1}_{r_1}|^2 = |a_{r_1,\alpha_1}|^2, \quad r_1 \in [m].$$

Sampling takes $\mathcal{O}(m)$ operations. At stage 2 for fixed r_1, we need to sample r_2 from the pmf $\phi(r_2|r_1,\boldsymbol{\alpha})$. But since $\phi(r_2|r_1,\boldsymbol{\alpha}) = e_2/d_2$ and d_2 does not involve r_2, we can sample from the pmf that is proportional to e_2 considered simply as a function of r_2 with r_1 fixed. Since e_2 is proportional to $|\,\mathrm{Per}\, A^{\alpha_1,\alpha_2}_{r_1,r_2}|^2$ this involves calculating m permanents of 2 × 2 matrices and then sampling; a further $\mathcal{O}(m)$ operations.

At stage k with $r_1,\dots,r_{k-1}$ already determined, we need to sample r_k from the pmf proportional to $|\,\mathrm{Per}\, A^{\alpha_1,\dots,\alpha_k}_{r_1,\dots,r_k}|^2$ considered simply as a function of r_k. Let $B = A^{\alpha_1,\dots,\alpha_k}_{r_1,\dots,r_k}$ for some arbitrary value of r_k, i.e. $b_{i,j} = a_{r_i,\alpha_j}$ for $i,j = 1,\dots,k$. Using the Laplace expansion and taking squared moduli

$$\left|\mathrm{Per}\, A^{\alpha_1,\dots,\alpha_k}_{r_1,\dots,r_k}\right|^2 = \left|\sum_{\ell=1}^{k} a_{r_k,\alpha_\ell}\,\mathrm{Per}\, B^{\diamond}_{k,\ell}\right|^2, \ r_k \in [m], \tag{4}$$

where $B^{\diamond}_{k,\ell}$ is defined in the statement of Lemma 2. From Lemma 2 all the values $\{\mathrm{Per}\, B^{\diamond}_{k,\ell}\}$ can be computed in $\mathcal{O}(k2^k)$ time. An array of length m is then obtained from (4) by summing k terms for each $r_k \in [m]$. Taking a single sample from the pmf proportional to the array takes $\mathcal{O}(m)$ time. This gives a total time complexity for stage k of $\mathcal{O}(k2^k)$ + $\mathcal{O}(mk)$.

The total time for all stages $k = 1, \ldots, n$ is therefore $\mathcal{O}(n2^n) + \mathcal{O}(mn^2)$. In practical terms, the exponentially growing term is approximately equivalent to the cost of evaluating two $n \times n$ permanents. The space usage is dominated by storing a single unnormalised pmf of size $\mathcal{O}(m)$ prior to sampling. □

Remarks: At the final stage when r_n is selected the value of $|\mathrm{Per}\, A^{\alpha_1,\ldots,\alpha_n}_{r_1,\ldots,r_n}|^2 = |\mathrm{Per}\, A_{\mathbf{r}}|^2$ will have been computed as in (4). Hence $q(\mathbf{r})$ the pmf of the sample $\mathbf{r}$ is available at no extra computational cost. Finally note that when only a single value is to be sampled from $q(\mathbf{r})$ and since the columns of a Haar random unitary matrix are already in a random order, the random permutation $(\alpha_1, \ldots, \alpha_n)$ in Algorithm B can be taken to be $(1, 2, \ldots, n)$

4 Acknowledgements

RC was funded by EPSRC Fellowship EP/J019283/1. Thank you to Ashley Montanaro, Alex Neville, Anthony Laing and members of the Bristol QET Labs for introducing us to the Boson Sampling problem and for a number of detailed and helpful conversations.

Appendix

Constant factor overheads in Lemma 2: To compare the constant factor overheads in Lemma 2, we start by looking at $\mathrm{Per}\, B$. From Glynn's formula we have

$$\mathrm{Per}\, B = \frac{1}{2^{k-1}} \sum_{\delta} \left(\prod_{i=1}^{k} \delta_i \right) \prod_{j \in [k]} v_j^*(\delta),$$

where $\delta \in \{-1, 1\}^k$ with $\delta_1 = 1$ and $v_j^*(\delta) = \sum_{i=1}^{k} \delta_i b_{ij}$. As we work through the values of δ in Gray code order, updating $v_j^*(\delta), j \in [k]$ requires k additions. Evaluating the product requires $k - 1$ multiplications and there is one further addition as the sum is accumulated. There are 2^{k-1} steps making a total of $2k2^{k-1} = k2^k$ operations. For the calculation of $\{\mathrm{Per}\, B^{\diamond}_{k,\ell}, \ell \in [k]\}$ in Lemma 2, there are k additions for updating $v_j(\delta), j \in [k]$, $k - 1$ multiplications for calculating $\prod_{j \in [k]} v_j(\delta)$, k divisions for $\prod_{j \in [k]} v_j(\delta)/v_\ell(\delta), \ell \in [k]$ and a further k additions as the k partial sums are accumulated. So each of the 2^{k-2} steps requires $4k - 1$ operations. Consequently the computation time is the same as that of $\mathrm{Per}\, B$ to the first order of approximation. In practice addition and multiplication with complex arithmetic will have different costs in terms of floating point operations but these are balanced in the two calculations.

Collision probability: Suppose that $\mathbf{z}$ is sampled from $q(\mathbf{z})$ in (2) and that $\mathbf{s}$ is the associated array of counts. The probability of duplicate elements in $\mathbf{z}$ is the probability that one of the elements of $\mathbf{s}$ is larger than 1. In general this probability will depend on the underlying Haar random unitary A.

From Lemma 1 we have

$$\begin{aligned} \mathbb{E}(s_i) &= \sum_{k=1}^{n} |a_{i,k}|^2 \\ \mathbb{E}(s_i(s_i - 1)) &= \sum_{c \in \mathcal{C}_2} \left|\mathrm{Perm}\, A^{c}_{i,i}\right|^2 = 4 \sum_{k<\ell} |a_{i,k} a_{i,\ell}|^2 \end{aligned} \qquad (5)$$

Let us denote the A-specific duplication probability, i.e. the chance of seeing duplicated values in $\mathbf{z}$, by P_D. We have from (5)

$$P_D \leqslant \sum_{i=1}^{m} \mathbb{P}(s_i \geqslant 2) \leqslant \frac{1}{2} \sum_{i=1}^{m} \mathbb{E}\, s_i(s_i - 1) = 2 \sum_{i=1}^{m} \sum_{k<\ell} |a_{i,k} a_{i,\ell}|^2.$$

Note that the expected value of the upper bound is $n(n-1)/(m+1) \sim n^2/m$ when averaged over A from the Haar random unitary class. See Aaronson and Arkhipov (2011, Theorem C.4) and Arkhipov and Kuperberg (2012) for comparison.

References

Aaronson, S. and Arkhipov, A. (2011). The computational complexity of linear optics. In *STOC '11: Proc. 43rd Annual ACM Symp. Theory of Computing*, pages 333–342.

Aaronson, S. and Arkhipov, A. (2014). BosonSampling is far from uniform. *Quantum Information & Computation*, 14(15-16):1383–1423.

Arkhipov, A. and Kuperberg, G. (2012). The bosonic birthday paradox. *Geometry & Topology Monographs*, 18:1–7.

Bentivegna, M., Spagnolo, N., Vitelli, C., Flamini, F., Viggianiello, N., Latmiral, L., Mataloni, P., Brod, D. J., Galvão, E. F., Crespi, A., et al. (2015). Experimental scattershot boson sampling. *Science advances*, 1(3):e1400255.

Björklund, A. (2016). Below all subsets for some permutational counting problems. In *15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016)*, pages 1–17.

Broome, M. A., Fedrizzi, A., Rahimi-Keshari, S., Dove, J., Aaronson, S., Ralph, T. C., and White, A. G. (2013). Photonic boson sampling in a tunable circuit. *Science*, 339(6121):794–798.

Crespi, A., Osellame, R., Ramponi, R., Brod, D. J., Galvão, E. F., Spagnolo, N., Vitelli, C., Maiorino, E., Mataloni, P., and Sciarrino, F. (2013). Integrated multimode interferometers with arbitrary designs for photonic boson sampling. *Nature Photonics*, 7(7):545–549.

Efraimidis, P. S. (2015). Weighted random sampling over data streams. In *Algorithms, Probability, Networks, and Games*, pages 183–195. Springer.

Feller, W. (1968). *An introduction to probability theory and its applications. - Vol. 1.* Wiley.

Glynn, D. G. (2010). The permanent of a square matrix. *European Journal of Combinatorics*, 31(7):1887–1891.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *STOC '00: Proc. 28th Annual ACM Symp. Theory of Computing*, pages 212–219.

Hastings, W. K. (1970). Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Kronmal, R. A. and Jr., A. V. P. (1979). On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218.

Latmiral, L., Spagnolo, N., and Sciarrino, F. (2016). Towards quantum supremacy with lossy scattershot boson sampling. *New Journal of Physics*, 18(11).

Lund, A., Bremner, M. J., and Ralph, T. (2017). Quantum sampling problems, BosonSampling and quantum supremacy. *arXiv:1702.03061*.

Marcus, M. and Minc, H. (1965). Permanents. *The American Mathematical Monthly*, 72(6):577–591.

Neville, A., Sparrow, C., Clifford, R., Johnston, E., Montanaro, A., and Laing, A. (2017). No imminent quantum supremacy for boson sampling. *arXiv:1705.00686*.

Nijenhuis, A. and Wilf, H. S. (1978). *Combinatorial algorithms: for computers and calculators*. Academic press.

Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1&2):223–252.

Ryser, H. J. (1963). *Combinatorial mathematics*, volume 14 of *Carus mathematical monographs*. The Mathematical Association of America.

Shchesnovich, V. (2013). Asymptotic evaluation of bosonic probability amplitudes in linear unitary networks in the case of large number of bosons. *International Journal of Quantum Information*, 11(05):1350045.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.

Spagnolo, N., Vitelli, C., Bentivegna, M., Brod, D. J., Crespi, A., Flamini, F., Giacomini, S., Milani, G., Ramponi, R., Mataloni, P., et al. (2014). Experimental validation of photonic boson sampling. *Nature Photonics*, 8(8):615–620.

Spring, J. B., Metcalf, B. J., Humphreys, P. C., Kolthammer, W. S., Jin, X.-M., Barbieri, M., Datta, A., Thomas-Peter, N., Langford, N. K., Kundys, D., et al. (2013). Boson sampling on a photonic chip. *Science*, 339(6121):798–801.

Tichy, M. C. (2011). *Entanglement and Interference of Identical Particles*. PhD thesis, Freiburg University.

Tichy, M. C., Mayer, K., Buchleitner, A., and Mølmer, K. (2014). Stringent and efficient assessment of boson-sampling devices. *Physical review letters*, 113(2):020502.

Tillmann, M., Dakić, B., Heilmann, R., Nolte, S., Szameit, A., and Walther, P. (2013). Experimental boson sampling. *Nature Photonics*, 7(7):540–544.

Valiant, L. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189 – 201.

Walker, A. J. (1974). New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128.

Walschaers, M., Kuipers, J., Urbina, J.-D., Mayer, K., Tichy, M. C., Richter, K., and Buchleitner, A. (2016). Statistical benchmark for bosonsampling. *New Journal of Physics*, 18(3):032001.

Wang, H., He, Y., Li, Y.-H., Su, Z.-E., Li, B., Huang, H.-L., Ding, X., Chen, M.-C., Liu, C., Qin, J., Li, J.-P., He, Y.-M., Schneider, C., Kamp, M., Peng, C.-Z., Hoefling, S., Lu, C.-Y., and Pan, J.-W. (2016). Multi-photon boson-sampling machines beating early classical computers. *arXiv:1612.06956*.

Wang, S.-T. and Duan, L. (2016). Certification of boson sampling devices with coarse-grained measurements. *Bulletin of the American Physical Society*.

Wu, J., Liu, Y., Zhang, B., Jin, X., Wang, Y., Wang, H., and Yang, X. (2016). Computing Permanents for Boson Sampling on Tianhe-2 Supercomputer. *arXiv:1606.05836*.