

Laplace expansions and tree decompositions: A faster polytime algorithm for shallow nearest-neighbor boson sampling

Samo Novák^{1,2,3,4,*} and Raúl García-Patrón^{1,5,†}

¹Laboratory for the Foundations of Computer Science, School of Informatics, Informatics Forum, *University of Edinburgh*, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom

²Mathematical Institute, *University of Oxford*, Andrew Wiles Building, Radcliffe Observatory Quarter (550), Woodstock Road, Oxford OX2 6GG, United Kingdom

³ORCA Computing, LG, 30 Eastbourne Terrace, London W2 6LA, United Kingdom

⁴Inria Paris, 48 rue Barrault, 75013 Paris, France

⁵Phasecraft Limited, 77 Charlotte Street, London W1T 4PW, United Kingdom



(Received 14 January 2026; accepted 6 February 2026; published 16 March 2026)

In a boson sampling quantum optical experiment, we send n individual photons into an m -mode interferometer and measure the occupation pattern on the output. The statistics of this process depend on the permanent of a matrix representing the experiment, which is itself a #P-hard problem to compute, and this is the reason why ideal and fully general boson sampling is hard to simulate on a classical computer. We exploit the fact that, for a nearest-neighbor shallow circuit, i.e., depth $D = O(\log m)$, one can adapt the algorithm by Clifford and Clifford [SODA '18 (2018), pp. 146–155] to exploit the sparsity of the shallow interferometer using an algorithm by Cifuentes and Parrilo [Lin. Alg. Appl. **493**, 45 (2016)] that can efficiently compute a permanent of a structured matrix from a tree decomposition. Our algorithm generates a sample from a shallow circuit in time $O(n^2 2^\omega \omega^2) + O(\omega n^3)$, where ω is the treewidth of the decomposition that satisfies $\omega \leq 2D$ for nearest-neighbor shallow circuits. The key difference in our work with respect to previous work using similar methods is the reuse of the structure of the tree decomposition, allowing us to adapt the Laplace expansion used by Clifford and Clifford, which removes a significant factor of m from the running time, especially as $m > n^2$ is a requirement of the original boson sampling proposal.

DOI: [10.1103/PhysRevA.113.032428](https://doi.org/10.1103/PhysRevA.113.032428)

I. INTRODUCTION

Boson sampling was introduced by Aaronson and Arkhipov [1] in 2010 to propose a suitable candidate for the experimental demonstration of quantum advantage, which is when a quantum computer can efficiently solve a problem that a classical computer cannot [2]. Boson sampling consists of sending n fully indistinguishable photons through an interferometer acting on m modes and detecting the photons as they come out at the output of the interferometer. The work by Aaronson and Arkhipov provided strong evidence that such a quantum optics circuit would be hard to simulate on a classical computer when the interferometer was chosen at random from the Haar ensemble.

The intuition behind the hardness of boson sampling relies on the fact that its output probabilities require computing permanents of matrices that scale with the number of photons

in the interferometer. The permanent of a matrix $A \in \mathbb{C}^{n \times n}$ is defined as

$$\text{per } A = \sum_{\sigma \in S_n} \prod_{i=1}^n A_{i, \sigma(i)},$$

where σ is a permutation from the symmetric group on n elements S_n . Despite its similarity with the definition of the determinant (without switching signs), computing the permanent exactly or approximating it for matrices with complex entries is #P-hard [3]. The best-known algorithms have exponential running time of $O(n2^n)$: these are due to Ryser [4] and Glynn [5].

A celebrated theoretical result was the sampling algorithm by Clifford and Clifford that can generate an n -photon boson sampling output with a computational cost that is, up to a factor of 2, equivalent to the computation of the permanent of a matrix of size n . Specifically, the running time of their algorithm is $O(n2^n) + O(mn^2)$ [6].

The prospect of performing an experimental demonstration of quantum advantage attracted a lot of experimental effort and further theoretical work, which recently led to experiments claiming they have reached quantum advantage on photonic platforms [7,8]. It is known that one can design classical algorithms that exploit the presence of imperfections in the hardware, such as losses and distinguishability of photons,

*Contact author: samo.novak@inria.fr

†Contact author: rgarcia3@exseed.ed.ac.uk

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

to run faster and potentially challenge the quantum advantage claims. For example, it was known that, for integrated photonic circuits composed of nearest-neighbor gates with transmission rates decaying exponentially with the depth of the circuit, circuits of depth $D = \Omega(\log m)$ may be simulated as thermal noise in polynomial time, and that a noisy shallow circuit can be simulated in quasipolynomial time using tensor networks [9].

The fact that simulating a shallow circuit [i.e., with depth $D = O(\log m)$] requires quasipolynomial time was expected to be a feature of the use of tensor network techniques, rather than a fundamental result. It was believed that one could find a polynomial time algorithm; for example, using results about efficient computation of permanents of sparse matrices by Cifuentes and Parrilo [10]. Indeed, this combination was used by Oh *et al.* in Ref. [11] to prove some complexity results about the (non)-hardness of boson sampling circuits with a lattice structure. Furthermore, an efficient algorithm for shallow Gaussian boson sampling circuits, a variant of boson sampling using squeezed states instead of single photons, was proven by Qi *et al.* in Ref. [12] using similar ideas.

In this work, we revisit the initial conjecture in Ref. [9] and show how one can use the result of Cifuentes and Parrilo [10] to design a polynomial-time algorithm for boson sampling from shallow circuits, exploiting the fact that the computation of a permanent scales polynomially in the treewidth of the graph corresponding to our photonic circuit of interest. A similar approach was given by Oh *et al.* [11], who gave the running time of $O(mn^2 2^\omega \omega^2)$.

We present an algorithm that reproduces all the key advantages of the initial Clifford and Clifford algorithm in Ref. [6], adapted to the nearest-neighbors shallow circuit scenario. Our main technical contribution is the following: We adapt the use of the sparsity of matrices in the algorithm by Cifuentes and Parrilo [10] to combine it with the use of the Laplace expansion in the algorithm by Clifford and Clifford. We do this in a time-efficient manner to remove the dependency on the number of modes in Ref. [11], achieving a running time of $O(n^2 2^\omega \omega^2) + O(\omega n^3)$.

A. Informal summary

Here, and in the running example of Fig. 1, we give an informal summary of the present work. A measurement outcome of a boson sampling experiment, defined by giving the interferometer and input and output states [see Fig. 1(a)], can be represented by a bipartite graph encoding the connectivity of the interferometer [see Fig. 1(b)]. Here, the vertices in the two partitions correspond to input (respectively, output) photons, and weighted edges correspond to nonzero transition amplitudes between individual photon positions. The probability amplitude of measuring the whole outcome of several indistinguishable photons is the sum over the perfect matchings in this graph, i.e., the permanent. In the shallow regime, the graph in Fig. 1(b) is sparse, and we capture this sparsity using a tree decomposition in Fig. 1(c), where we group edges of the graph into nodes of a new tree structure, subject to simple connectivity conditions. The algorithm of Cifuentes

and Parrilo exploits this tree decomposition to compute the permanent efficiently using a dynamical program [10].

Our main technical contribution is shown in Fig. 1(d) of the running example. We adapt the use of the Laplace expansion of the permanent, a vital component of the algorithm by Clifford and Clifford [6], to the permanent computation using tree decompositions. Computing the probability of the k th photon at a given output r_k [“•” in Fig. 1(a)], given a pattern of input and output modes for the previous $k - 1$ photons (“•”), is captured by the permanent $\text{per}\mathcal{W}^{(r_k)}$ of a matrix where only the last row depends on the choice of the new possible photon output r_k . Clifford and Clifford [6] use the Laplace expansion:

$$\text{per}\mathcal{W}^{(r_k)} = \sum_{j=1}^k \mathcal{W}_{k,j}^{(r_k)} \text{per}\mathcal{W}_{\diamond(k,j)}^{(r_k)}, \quad (1)$$

where $\mathcal{W}_{\diamond(k,j)}^{(r_k)}$ is the submatrix of $\mathcal{W}^{(r_k)}$ with row k and column j removed. As we remove the last row k , these matrices are independent of the choice of r_k . Clifford and Clifford exploit this fact to precompute the values of $\{\text{per}\mathcal{W}_{\diamond(k,j)}^{(r_k)}\}_j$ to later separately do a simple calculation of the m possible output probabilities in polynomial time.

A naive approach to adapt the Cifuentes and Parrilo algorithm for nearest-neighbor shallow circuits would require a new tree decomposition for each of these submatrices, which degrades performance. However, by ensuring that the nodes of the tree decomposition we use are in one-to-one correspondence to input photon vertices, and that the tree is linear (no branching, i.e., isomorphic to a path graph), we can compute each summand in the Laplace expansion by only a small modification of the original tree decomposition. As shown in Fig. 1(d), we formalize this as temporarily removing a node and replacing it with an “empty” dummy for each removed input photon, although the dummy may need some minimal contents. We may conceptualize this as a “machine head” walking the tree from one end to the other, at each step replacing the node it visits, and computing the permanent $\text{per}\mathcal{W}_{\diamond(k,j)}^{(r_k)}$ in (1). These permanents together form the Laplace decomposition, and this order of computing then allows us to reuse almost all dynamic programming tables required by the Cifuentes and Parrilo algorithm.

We remark that our abundant reuse of a single global tree decomposition requires us to work in the noncollision setting of $m = \Omega(n)$: a collision of output photons introduces a cycle in the relevant graph and invalidates the global tree decomposition.

B. Structure of the paper

In Sec. II, we give a short introduction to linear optics, boson sampling, and the Clifford and Clifford algorithms. We offer a more pedagogical overview in Appendixes A and B. In Sec. III, we introduce the tree decomposition of matrices and the algorithm of Cifuentes and Parrilo for computing permanents of sparse matrices. This is complemented by Appendix C.

The construction of our new algorithm takes place in Sec. IV, where we present some necessary constructions

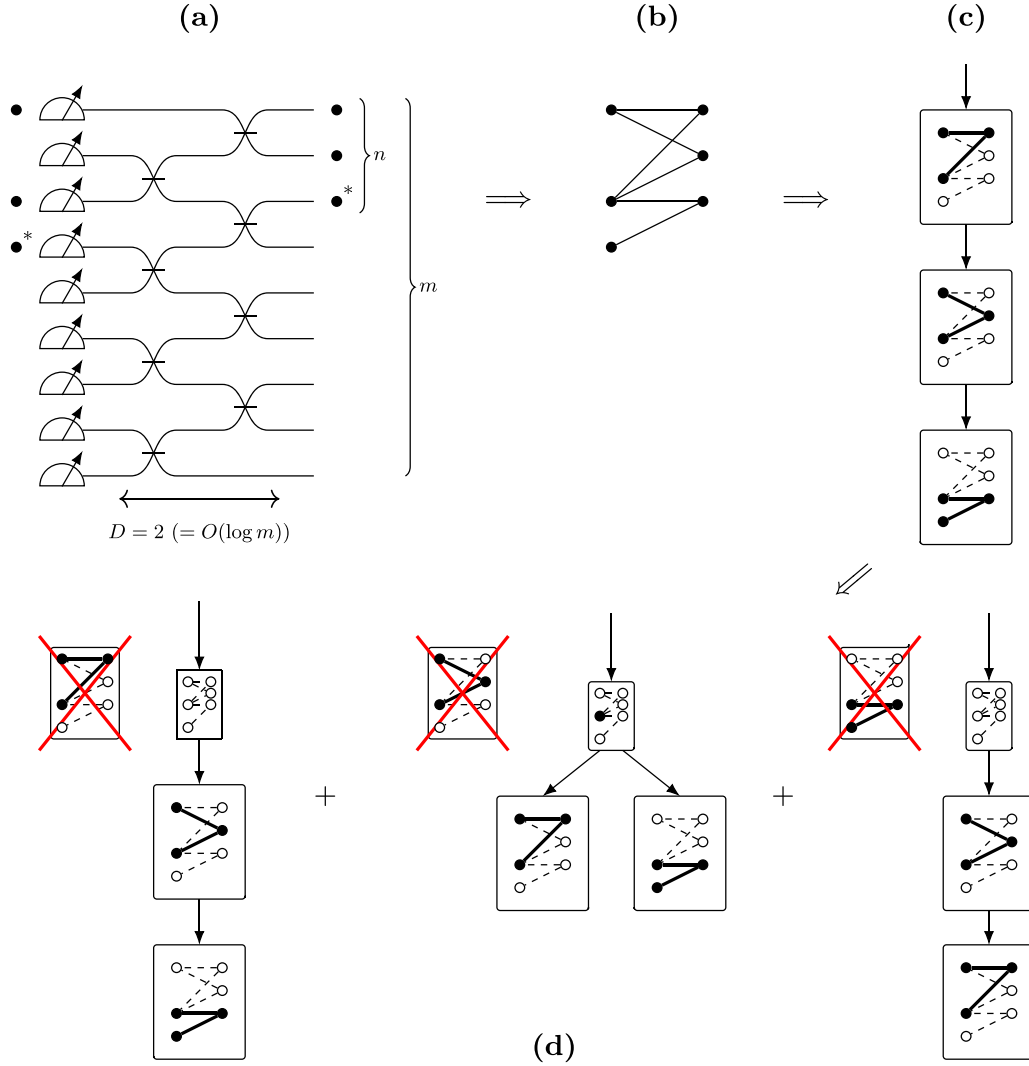


FIG. 1. Overview of our work using an example: (a) The shallow brickwork circuit of nearest-neighbor beam splitters with inputs on the right and outputs on the left, showing one measurement outcome with previously sampled photons “•” and a new photon “•*” added in this step of the Clifford and Clifford algorithm to expand the sample. (b) A weighted bipartite graph represents the outcome: vertices are input and output photons, and nonzero transition amplitudes between them give weighted edges. The outcome amplitude is the sum of weighted perfect matchings in this graph, i.e., the permanent. (c) To exploit the sparsity structure, we construct a tree decomposition: a tree whose nodes contain sets of edges of the graph in panel (b), subject to simple rules. We use a linear decomposition (no branching) where each node corresponds uniquely to an input photon vertex. We then modify the tree decomposition in several steps, each time temporarily removing a different input photon—column—node of the tree, replacing the latter by a dummy with no new information, seen in panel (d). Permanents of these modified trees form the Laplace expansion, our main technical contribution. When done in the right order, conceptualized as a “machine head” walking the tree from one end to the other, we can reuse most of the dynamic programming tables required by the Cifuentes and Parrilo permanent algorithm. As seen in the middle term, we note that the dummy may need to be nonempty.

representing shallow photonic circuits, and in Sec. V, where we present the algorithm itself. We conclude and discuss open problems in Sec. VI, and we expand on some technical details in Appendixes D and E.

II. BOSON SAMPLING AND CLIFFORD AND CLIFFORD ALGORITHM

To understand our new algorithm, we first review important concepts of single-photon boson sampling and the Clifford and Clifford algorithms. A more pedagogical exposition and detailed derivations are found in Appendixes A and B.

A. Short introduction to single-photon boson sampling

In an ideal boson sampling circuit, the input and output can be written in a basis of quantum states corresponding to the location of n photons in m modes. These can be represented as number (Fock) states $|\underline{n}\rangle = |n_1, \dots, n_m\rangle \in (\mathbb{C}^n)^{\otimes m}$, where $n_i \in \mathbb{N}$ is the number of photons in mode i , and $\sum_i n_i = n$. In this paper, however, it will be useful to see these states in the first-quantization representation, where every photon corresponds to a qudit of dimension m . Its logical information encodes the mode in which the photon is located, i.e., a basis state $|i\rangle$ means a photon in mode i . We ensure the indistinguishability of photonic states by writing the global quantum

state as a symmetric superposition over all permutations of the qudits:

$$|\Phi_{\underline{n}}\rangle = \frac{1}{\sqrt{n! \prod_{i=1}^m n_i!}} \sum_{\sigma \in S_n} \sigma |\underline{z}\rangle, \quad (2)$$

where we write $|\underline{z}\rangle := |1\rangle^{\otimes n_1} \otimes |2\rangle^{\otimes n_2} \otimes \dots \otimes |m\rangle^{\otimes n_m}$ with the qudit factors sorted in nondescending order, where σ is a permutation in the symmetric group on n elements S_n , and the prefactor on the right-hand side (RHS) is its normalization [13]. By convention, we denote the sorted product states as $|\underline{z}\rangle$ and various reorderings by permutations as $|\underline{r}\rangle$.

An interferometer is a linear optical device that is represented by a unitary matrix $\mathcal{U} \in \mathbb{U}(m)$ acting linearly on a single-qudit basis state $|i\rangle$ as

$$|i\rangle \xrightarrow{\mathcal{U}} \sum_{j=1}^m \mathcal{U}_{i,j} |j\rangle \equiv \mathcal{U}|i\rangle. \quad (3)$$

On a basis state of n photons:

$$|\underline{z}\rangle \equiv \bigotimes_{k=1}^n |z_k\rangle \xrightarrow{\mathcal{U}} \bigotimes_{k=1}^n (\mathcal{U}|z_k\rangle) = \mathcal{U}^{\otimes n} |\underline{z}\rangle, \quad (4)$$

where we have introduced the notation $|z_k\rangle$ to mean the k th qudit in the product $|\underline{z}\rangle$. Note that the unitarity of $\mathcal{U}$ ensures that the total number of photons is preserved [14].

1. Outcome probabilities

For a given input state $|\underline{n}\rangle$ the output probability distribution of a boson sampling circuit of interferometer $\mathcal{U}$ is given by the Born rule:

$$\mathbb{P}[\underline{n}'|\underline{n}] = |\langle \Phi_{\underline{n}'} | \mathcal{U}^{\otimes n} | \Phi_{\underline{n}} \rangle|^2, \quad (5)$$

where $|\underline{n}'\rangle$ is the output of the photon number measurement, and where we have $\sum_i n'_i = \sum_i n_i = n$, as the photon number is preserved by ideal lossless circuits. Rewriting (5) in first quantization, we obtain

$$\mathbb{P}[\underline{n}'|\underline{n}] = \left| \frac{1}{\sqrt{\prod_{i=1}^m n_i! n'_i!}} \sum_{\pi \in S_n} \prod_{j=1}^n \mathcal{V}_{j,\pi(j)}^{\underline{n}',\underline{n}} \right|^2 \quad (6a)$$

$$= \frac{|\text{per} \mathcal{V}^{\underline{n}',\underline{n}}|^2}{\prod_{i=1}^m n_i! n'_i!}, \quad (6b)$$

where we define a matrix $\mathcal{V}^{\underline{n}',\underline{n}}$ to represent the transition amplitudes of individual photons in the $|\underline{n}\rangle \rightarrow |\underline{n}'\rangle$ process, in the qudit basis. Its components are $\mathcal{V}_{l,h}^{\underline{n}',\underline{n}} := \langle z'_l | \mathcal{U} | z_h \rangle$. In (6a), we recognize the permanent of $\mathcal{V}^{\underline{n}',\underline{n}}$. We provide a complete derivation in Appendix A 2.

2. Graphical representation of $\mathcal{V}^{\underline{n}',\underline{n}}$

While $\mathcal{U} \in \mathbb{C}^{m \times m}$ represents an operator acting on modes, the matrix $\mathcal{V}^{\underline{n}',\underline{n}} \in \mathbb{C}^{n \times n}$ encodes the behavior of the n photons as they pass through the interferometer. In general, we get $\mathcal{V}^{\underline{n}',\underline{n}} \in \mathbb{C}^{n \times n}$ by taking n'_i copies of row i in $\mathcal{U}$, creating an $n \times m$ matrix, and from this n_j copies of column j [13]. In particular, unoccupied modes, or photon collisions, lead to absent or copied rows and columns, respectively.

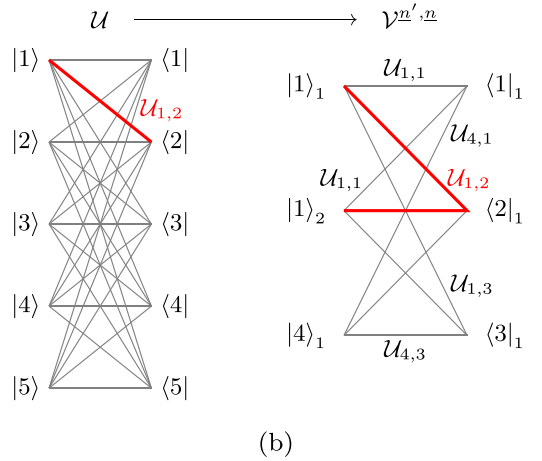
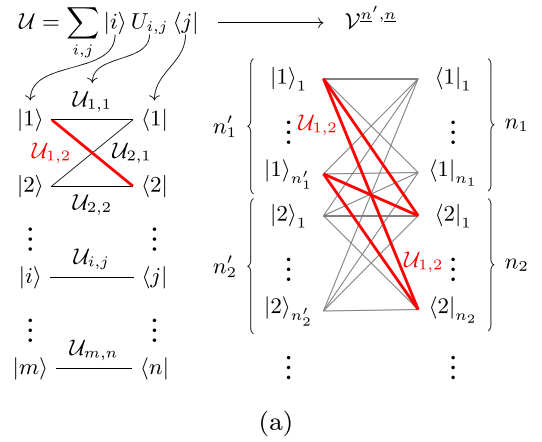


FIG. 2. Graphical representation: (a) Graphs $G(\mathcal{U})$ and $G(\mathcal{V}^{\underline{n}',\underline{n}})$ and their relationship. Left (right) partition corresponds to outputs (inputs). Note that $\mathcal{U}$ is the biadjacency matrix of $G(\mathcal{U})$, and likewise $\mathcal{V}^{\underline{n}',\underline{n}}$ of $G(\mathcal{V}^{\underline{n}',\underline{n}})$. The output partition of the graph $G(\mathcal{V}^{\underline{n}',\underline{n}})$ consists of n'_i copies $|i\rangle_1, \dots, |i\rangle_{n'_i}$ of output vertex $|i\rangle$ of $G(\mathcal{U})$; inputs are similarly copied. If there is an edge $e = (|i\rangle, \langle j|)$ in $G(\mathcal{U})$, then $G(\mathcal{V}^{\underline{n}',\underline{n}})$ has edges $(|i\rangle_a, \langle j|_b)$ for all a, b , and these have the same weight as e : We show in red the edge $(|1\rangle, \langle 2|)$ of $G(\mathcal{U})$ and its copies in $G(\mathcal{V}^{\underline{n}',\underline{n}})$, all with weights $\mathcal{U}_{1,2}$. Some edges are omitted for clarity. (b) Example: $n = n' = 3$, $m = 5$, input $|\underline{n}\rangle = |1, 1, 1, 0, 0\rangle$, and output $|\underline{n}'\rangle = |2, 0, 0, 1, 0\rangle$.

To facilitate the understanding of the graphical methods used in Secs. III to V, we fruitfully interpret both $\mathcal{U}$ and $\mathcal{V}^{\underline{n}',\underline{n}}$ as weighted adjacency matrices of bipartite graphs, $G(\mathcal{U})$ and $G(\mathcal{V}^{\underline{n}',\underline{n}})$, respectively, as shown in Fig. 2, with a particular labeling of vertices useful to understand the transition from $\mathcal{U}$ to $\mathcal{V}^{\underline{n}',\underline{n}}$. The graph $G(\mathcal{U})$ encodes the relation from input to output modes of the linear interferometer. Vertices $|i\rangle$ ($\langle j|$) correspond to output (input) modes, and edges to nonzero probability input-output transitions. The graph $G(\mathcal{V}^{\underline{n}',\underline{n}})$ has as vertices copies of $|i\rangle$, $\langle j|$ from $G(\mathcal{U})$ labeled $|i\rangle_k$, $\langle j|_l$. Edges are likewise copied from $G(\mathcal{U})$, along with their weights, and they represent photons traveling between modes in the $|\underline{r}\rangle$ basis. The graph represents the possible paths that all the distinguishable photons may take from input $|\underline{z}\rangle$ to output $|\underline{z}'\rangle$. Each choice of which input photon becomes which output photon corresponds to a perfect matching in $G(\mathcal{V}^{\underline{n}',\underline{n}})$, and the

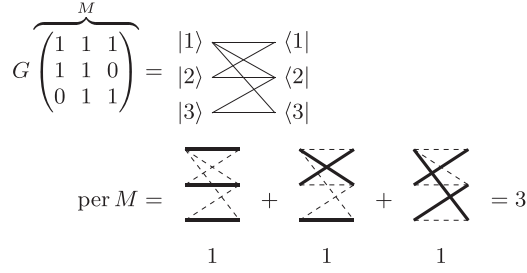


FIG. 3. Relation between $G(M)$ and $\text{per}M$ for an example binary matrix M : $\text{per}M$ counts the perfect matchings of $G(M)$.

amplitude of that matching is the product of the amplitudes of its edges. The total amplitude of $|\underline{z}\rangle \rightarrow |\underline{z}'\rangle$ (representing $|\underline{n}\rangle \rightarrow |\underline{n}'\rangle$) is the sum over all matchings, giving the permanent $\text{per}\mathcal{V}^{\underline{n}',\underline{n}}$. We show a simplified example of the relation to the permanent in Fig. 3.

B. Clifford and Clifford algorithms

The work [6] by Clifford and Clifford presents three conceptual steps to arrive at the current best general algorithm with running time $O(n2^n)$. In the present summary, and in our pedagogical exposition in Appendix B, we refer to these as Algorithms CC-A, CC-B, and CC-C. Note that [6] presents CC-B and CC-C together as a single algorithm. In this section, we focus on the transition from CC-B to the faster CC-C, which notably uses the crucial step of Laplace expansion of the permanent, adapting which allows us to obtain our algorithm in Sec. V.

1. Algorithms CC-A and CC-B

The idea behind CC-A consists of sampling in the $|\underline{r}\rangle$ basis (see Sec. II A) using a chain rule procedure that builds the sample one photon at a time. We work as if the photons were distinguishable by assigning them labels, and we compute marginal probabilities of partial samples $(r_1, \dots, r_k)$. The trick is to create a sample of one photon, commit to it, then enlarge to two photons, etc. Marginals of partial samples involve summing over the choices of input photons that are sampled, adding complexity. This is sped up in CC-B by permuting the input photons by a random, uniformly distributed $\alpha \in S_n$, new for each sample. As shown in Ref. [6], this gives the correct sampling distribution over several samples.

The marginals in CC-B require the computation of $\text{per}\tilde{\mathcal{V}}^{(r_1, \dots, r_k), \alpha([k])}$ where the matrix fulfils the same function as $\mathcal{V}^{\underline{n}', \underline{n}}$, but it represents a partial sample and is indexed in qudit basis. Its components are $\tilde{\mathcal{V}}_{i,j}^{(r_1, \dots, r_k), \alpha([k])} = \langle r_i | \mathcal{U} | \alpha(j) \rangle$: observe that $\alpha([k]) = \{\alpha(1), \dots, \alpha(k)\}$ selects the input modes and $(r_1, \dots, r_k)$ the outputs. We explain the steps to obtain CC-A and CC-B, as well as the running $O(mn2^{n+1})$ of CC-B, in Appendix B.

2. Laplace expansion: Algorithm CC-C

The final improvement of Ref. [6] transfers the factor m from the dominant exponential term to a less relevant quadratic term, leading to an $O(n2^n)$ algorithm that generates a sample with a factor of 2 of the cost of computing a single

output probability. This result is achieved using the Laplace expansion of the permanent, defined below, and this is the idea we adapt to tree decomposition techniques in Sec. V. We summarize the algorithm CC-C as pseudocode in Fig. 10 in Appendix B 3.

When expanding the sample from $k-1$ to k photons, we need to compute m permanents of matrices $\tilde{\mathcal{V}}^{(r_1, \dots, r_k), \alpha([k])}$, where all k columns are shared (predetermined input photons), and all rows except r_k are also the same (previously sampled output photons). We simplify the notation and denote $\mathcal{W} = \tilde{\mathcal{V}}^{(r_1, \dots, r_{k-1}), \alpha([k])}$ the $(k-1) \times k$ matrix containing the shared rows and columns, and denote by $\mathcal{W}^{(r_k)} = \tilde{\mathcal{V}}^{(r_1, \dots, r_k), \alpha([k])}$ the matrix obtained by adding the row r_k of $\mathcal{U}$ to $\mathcal{W}$. Using the Laplace expansion of the permanent, we write

$$\text{per}\mathcal{W}^{(r_k)} = \sum_{j=1}^k \mathcal{W}_{k,j}^{(r_k)} \text{per}\mathcal{W}_{\diamond(k,j)}^{(r_k)} \quad (7a)$$

$$= \sum_{j=1}^k \mathcal{U}_{r_k, \alpha(j)} \text{per}\mathcal{W}_{\diamond j}. \quad (7b)$$

In (7a), $\mathcal{W}_{\diamond(k,j)}^{(r_k)}$ is the $(k-1) \times (k-1)$ submatrix of $\mathcal{W}^{(r_k)}$ with the row k and column j removed. By construction, the matrices $\mathcal{W}_{\diamond(k,j)}^{(r_k)}$ are the same for all choices of r_k . In (7b), we replace this by $\mathcal{W}_{\diamond j}$ denoting the $(k-1) \times (k-1)$ submatrix of $\mathcal{W}$ with column j removed.

The final improvement by Clifford and Clifford is an adaptation of Glynn's permanent algorithm [5] to compute the family $\{\text{per}\mathcal{W}_{\diamond j}\}_{j=1}^k$ in combined time $O(k2^k)$ and space $O(k)$. This is the same scaling, up to a constant, as that required to compute the permanent of the whole $k \times k$ matrix. Once these permanents are known, computing the m output probabilities requires only the use of the Laplace expansion (7b), taking $O(mk)$ operations. We elaborate on the time complexity in Appendix C 2.

III. CIFUENTES AND PARILLO ALGORITHM

We now present the concept of tree decompositions of a graph and its use in the algorithm by Cifuentes and Parrilo (CP) to compute the permanent of sparse matrices using dynamic programming [10]. We adapt this to CC-C in Sec. V.

A. Tree decompositions

A tree decomposition is a structure that captures how (dis)connected a graph is, and it can, under some conditions, speed up the computation of solutions of instances of problems that are otherwise worst-case NP-hard [15]. Finding an optimal tree decomposition can itself be NP-hard, however, there exist good heuristics that allow improvements for practical problems. An example where tree decomposition is used by the quantum information community is the optimization of tensor network contractions. In our case, the Cifuentes and Parrilo algorithm exploits tree decompositions to compute permanents of structured matrices faster than can be done by using the standard algorithms by Ryser or Glynn.

Let G be a bipartite graph of a matrix, in the sense of Sec. II A 2, where the set of vertices $V = \mathcal{R} \sqcup \mathcal{C}$ consists of row and column labels ($\mathcal{R}, \mathcal{C}$, respectively). The edges

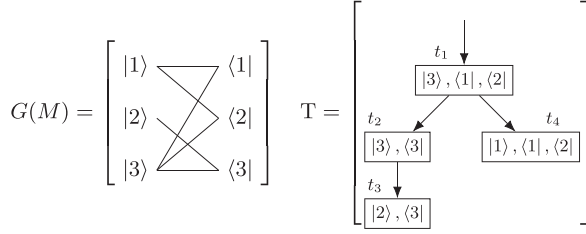


FIG. 4. Example tree decomposition T of some graph of matrix $G(M)$. This tree is rooted at t_1 .

$E \subseteq \mathcal{R} \times \mathcal{C}$, each connecting one row and one column, correspond to the nonzero matrix elements. In the following, we use the notation 2^X for the powerset of a set X , i.e., the set of all subsets of X .

Definition III.1 (tree decomposition of a bipartite graph). Let G be a bipartite graph. A tree decomposition $T = (T, \rho, \kappa)$ of G is a triple consisting of a tree T rooted in $\text{root}(T)$, where for each node $t \in T$, we denote by $\text{ch}(t)$ the unordered set of its children; and the functions $\rho : T \rightarrow 2^{\mathcal{R}}$ and $\kappa : T \rightarrow 2^{\mathcal{C}}$ that label the nodes with subsets of $\mathcal{R}$ and $\mathcal{C}$. We say that a node $t \in T$ contains the elements of $\rho(t)$ and $\kappa(t)$. A tree decomposition must satisfy the following axioms:

(T1) The union of contents over the entire tree covers all vertices of the graph: $\rho(T) = \mathcal{R}$ and $\kappa(T) = \mathcal{C}$, where we denote $\rho(T) = \bigcup_{t \in T} \rho(t)$, and analogously $\kappa(T) = \bigcup_{t \in T} \kappa(t)$.

(T2) Every edge of the graph is contained in some node of the tree. This means that for each $(r, c) \in E$, there exists a node $t \in T$ that contains both its endpoints, i.e., $r \in \rho(t)$ and $c \in \kappa(t)$.

(T3) For every $r \in \mathcal{R}$ ($c \in \mathcal{C}$), the set of all nodes that contain it is a subtree of T , i.e., all nodes containing r (c) are connected in T .

We show an example tree decomposition in Fig. 4. By convention, we write the decomposition upright as T and the underlying tree slanted as T . Whenever we say the word node, we mean some $t \in T$ in a tree decomposition, and when we say vertex, it means some $v \in V$ of the graph G being studied. We write T_t for the subtree of T rooted at $t \in T$, that is, the tree that contains t and all its descendants. An important consequence of Axiom (T3) that will be relevant later is the following: if a vertex or an edge is contained in both T_c and $T_{c'}$, where c, c' are two distinct children of t , then it must also be contained in the parent node t .

1. Treewidth

For any graph G , there are generally many tree decompositions; we denote this collection $\mathcal{T}(G)$. However, not all of them are useful, e.g. the trivial case of a single node containing the entire graph G . The following definition captures how good a tree decomposition is:

Definition III.2 (treewidth [10,15]) For a tree decomposition $T = (T, \rho, \kappa)$, we define its treewidth as

$$\text{tw}(T) := \max_{t \in T} \{|\rho(t)| + |\kappa(t)|\} - 1, \quad (8a)$$

Algorithm CP: Bipartite permanent

Input:

- a matrix $M \in \mathbb{C}^{n \times n}$,
- a tree decomposition $T = (T, \rho, \kappa)$ of $G(M)$

Result: the permanent per M

```

1 order ← list of nodes in  $T$ , in topological order,
  starting from leaves
2 for  $t \in \text{order}$  do
3    $Q[t](R, C) \leftarrow \text{per } M|_{R, C}$  for  $R \subseteq \rho(t)$  and  $C \subseteq \kappa(t)$ 
4   if  $t$  is a leaf, i.e.  $\text{ch}(t) = \emptyset$  then
5      $P[t] \leftarrow Q[t]$ 
6   else
7     for  $c_j \in \text{ch}(t)$  do
8        $\begin{cases} \Delta_{c_j}^\rho \leftarrow \rho(c_j) \setminus \rho(t) \\ \Delta_{c_j}^\kappa \leftarrow \kappa(c_j) \setminus \kappa(t) \end{cases}$ 
9       for  $R \subseteq \Delta_{c_j}^\rho$  and  $C \subseteq \Delta_{c_j}^\kappa$  do
10         $Q'[t|c_j](R, C) \leftarrow (-1)^{|R|} \cdot Q[t](R, C)$ 
11         $Q''[t \leftarrow c_j](R, C) \leftarrow$ 
12           $P[c_j](R \cup \Delta_{c_j}^\rho, C \cup \Delta_{c_j}^\kappa)$ 
13         $P[t](R, C) \leftarrow$  subset convolution of  $Q[t]$  and
14          all  $Q'[t|c_j], Q''[t \leftarrow c_j]$  as in (9)
15 return  $P[r](\rho(r), \kappa(r)) = \text{per } M$  where  $r = \text{root}(T)$ 
```

FIG. 5. Pseudocode of the Cifuentes and Parrilo algorithm.

that is, the size of the largest node t in terms of its contents, minus one. For a graph G , we define its treewidth as

$$\widehat{\text{tw}}(G) := \min_{T \in \mathcal{T}(G)} \text{tw}(T), \quad (8b)$$

that is, the minimum treewidth of any possible tree decomposition of G . We distinguish the treewidth of a graph with a hat.

The treewidth measures how different a graph is from a tree, and we define the RHS of (8a) with -1 so that a tree Q has $\widehat{\text{tw}}(Q) = 1$.

Both in tensor network contraction and in our permanent computations, the running time and memory requirements scale exponentially with the treewidth of the graph. Computing with tree-like graphs is, in general, much easier than graphs with higher treewidth, which contain many cycles [15]. In our case, the intuition is that a graph with high treewidth, and thus many cycles, allows many perfect matchings, i.e., summands of the permanent, which leads to a more expensive computation.

B. Permanent algorithm

We now present the permanent algorithm by Cifuentes and Parrilo from Ref. [10, Algorithm 2], to which we refer as Algorithm CP, and whose pseudocode can be found in Fig. 5. The algorithm exploits the structure and sparsity of the matrix to accelerate the computation of its permanent. Using a tree decomposition of the graph of the matrix, it is able to decompose the matrix into smaller parts and compute their permanents. The tree structure then tells it how to join these parts together. It is a dynamic programming algorithm that

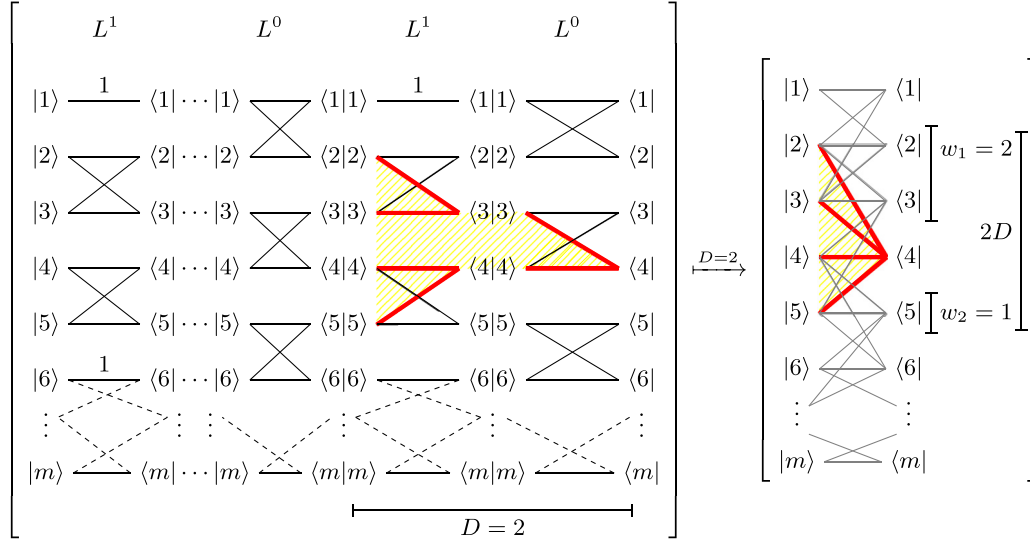


FIG. 6. An even-depth example (last layer L^1) of an alternating array of beam splitters in our architecture, represented using the graphical notation from Fig. 2. The columns represent the layers of gates L^0 and L^1 , and the crossings are beam splitters. For each mode, one beam splitter connects it with the mode above, and the next beam splitter with the mode below, or vice versa. The bold red lines show the possible paths of photons starting in mode 4, corresponding to its causal cone (filled yellow), over two layers of the circuit. The RHS is the corresponding graph $G(\mathcal{U})$ resulting from the first two layers of the circuit.

stores values in tables for later use. For an $n \times n$ matrix and a tree decomposition of its graph of treewidth ω , the running time is $O(n2^\omega \omega^2)$.¹

1. Dynamic programming tables

Algorithm CP builds the permanent of matrix M from the leaves upward. For all nodes and for the steps between child and parent nodes, we need to build two dynamic programming tables $Q[t](R, C)$ and $P[t](R, C)$, both indexed by a subset of rows $R \subseteq \rho(t)$ and columns $C \subseteq \kappa(t)$ belonging to the node t . The table Q stores the permanents of submatrices with rows and columns belonging to the node t , while the table P allows us to combine that information with the permanents of its children using a subset convolution:

$$P[t](R, C) = \sum \left(Q[t](R_t, C_t) \prod_{c_j \in \text{ch}(t)} Q'[t|c_j](R'_{c_j}, C'_{c_j}) \right. \\ \left. \times Q''[t \leftarrow c_j](R''_{c_j}, C''_{c_j}) \right). \quad (9)$$

Note that in the computation of $P[t]$, we also use two helper tables $Q'[t|c_j]$ and $Q''[t \leftarrow c_j]$ for each child c_j of t . These latter bring the values of $P[c_j]$ into the computation of $P[t]$. The summation runs over partitions of R (C) into a family of subsets $\{R_i^{(r, n)}\}_i$ ($\{C_i^{(r, n)}\}_i$). We expand on the computation of these tables and valid partitions in Appendix C 1. Note that the final result, the permanent of M , is found in the root node:

$$P[r](\rho(r), \kappa(r)) = \text{per} M. \quad (10)$$

¹The running time is conventionally given as $\tilde{O}(n2^\omega)$, which hides polynomial factors in ω , but the factor ω^2 can be recovered from [10, Lemma 2 and proof of Lemma 14] as detailed in Appendix C 2.

IV. PRELIMINARIES TO NEW ALGORITHM

Before we can construct our main contribution, an algorithm for polytime classical simulation of boson sampling from shallow interferometers, we need to review and define a few necessary constructions.

A. Nearest-neighbor circuits

We are interested in architectures of ideal interferometric circuits composed of nearest-neighbor interactions between modes. It is well known that such interferometers can be constructed from phase-shifters and beam splitters [16, 17], which are, respectively, one- and two-mode gates. The latter is defined as the following unitary acting on modes (cf. $\mathcal{U}$) [18]:

$$B(\theta, \phi_T, \phi_R) = \begin{pmatrix} e^{i\phi_T} \cos \theta & e^{i\phi_R} \sin \theta \\ -e^{-i\phi_R} \sin \theta & e^{-i\phi_T} \cos \theta \end{pmatrix}, \quad (11)$$

where θ is the coupling between the two modes, and ϕ_T (ϕ_R) is the transmitted (reflected) phase shift. Similarly to Ref. [19], we absorb the phase-shifters, being single-mode operations, into adjacent beam splitters whenever possible; thus, our circuits consist only of beam splitters.

We consider a construction with the beam splitters arranged in alternating layers, as displayed in Fig. 6, as in the Clements rectangular decomposition [20] that guarantees the universality of the interferometric circuit while minimizing its depth D . This decomposition also has practical advantages, as it guarantees more uniform losses among the different output modes, which limits the effects of those imperfections. A full circuit is composed of an alternation of two slightly different types of layers of beam splitters, L^0 and L^1 (where we omit parameters), that allow the spreading of the correlations over distant modes following a causal cone, emphasized in Fig. 6.

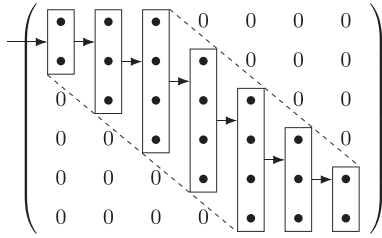


FIG. 7. Example linear tree decomposition T^C where tree nodes are shown as boxes encompassing possibly nonzero elements (“•”). Remark that nodes correspond to parts of columns of the matrix. The band is shown by dashed lines.

As a consequence of the causal cone, the unitary $\mathcal{U}$ of the circuit has a banded structure, meaning there exist numbers w_1, w_2 s.t. $\mathcal{U}_{i,j} = 0$ for $i - j > w_1$ or $j - i > w_2$. Therefore, the bandwidth w is bounded as

$$w = w_1 + w_2 + 1 \leq 2D, \quad (12)$$

as shown by Lemma D1 in Appendix D. This can be intuitively seen in Fig. 6, where the bandwidth corresponds to the furthest distance (in modes) a photon can travel away from its input mode. It is easy to see that each beam splitter layer expands each of w_1 and w_2 by at most 1, and the width after the first layer is 2.

B. Tree decomposition of a banded matrix

There are different tree decompositions one can design for an $m \times m$ banded matrix $\mathcal{U}$. For algorithmic design and simplicity, we use a tree decomposition $T^C = (T, \rho, \kappa)$ shown in Fig. 7 that is not optimal but simple.² The decomposition T^C consists of a linear tree $T = \{t_1, \dots, t_m\}$ where the node t_i has a single child t_{i+1} for $i = 1, \dots, m-1$ (and t_m is a leaf). There is a correspondence between the nodes and columns of the matrix, such that $\kappa(t_i) = \{i\}$ for all $i \in [m]$. This tree decomposition associates with each node t_i , corresponding to column i , all rows j where the matrix elements of that column may be nonzero: $|j\rangle \in \rho(t_i)$ if and only if $\mathcal{U}_{ji}$ is within the band. If $\mathcal{U}$ has bandwidths (w_1, w_2) , then in general

$$\rho(t_i) = \{|j - w_1\rangle, |j - w_1 + 1\rangle, \dots, |j + w_2\rangle\}, \quad (13)$$

modulo nonexistent row labels. There are at most w elements in $\rho(t_i)$, and together with a single element in $\kappa(t_i)$, recalling that the treewidth is defined with a constant -1 [see (8a)], this makes the treewidth of the decomposition $\text{tw}(T^C) \leq w \leq 2D$.

C. Tree decompositions of submatrices with column permutations

Having a low-width tree decomposition, we are able to compute the permanent of a matrix efficiently using

²The optimal decomposition uses two nodes per column, each containing $w - 1$ rows, where w is the number of rows with a nonzero in a column. These nodes are neighbors, one containing the upper w rows, and the other the lower. This was given by Cifuentes and Parrilo [10], modulo exchange of rows and columns between their paper and ours.

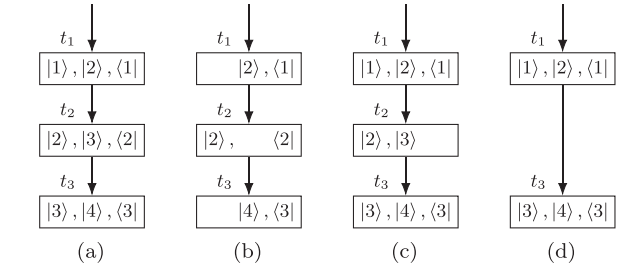


FIG. 8. Examples of tree restrictions: (a) A decomposition $T = T^C$. (b) A restriction $T|_{\mathcal{R}', \mathcal{C}'}$ to rows $\mathcal{R}' = \{|2\rangle, |4\rangle\}$ and columns $\mathcal{C}' = \{|1\rangle, |2\rangle, |3\rangle\}$. Vertices $|1\rangle$ and $|3\rangle$ are removed. (c) A different restriction, this time only removing the column $|2\rangle$, making the node t_2 redundant. It cannot contribute any new elements from the matrix. Instead, node t_2 is only used to make some values from table $P[t_3]$ available to t_1 . We can thus remove t_2 and connect t_1 directly to t_3 to obtain the equivalent decomposition $T \setminus t_2$ in panel (d).

Algorithm CP. However, while the original matrix $\mathcal{U}$ has a banded structure allowing us to find the decomposition, this structure is concealed when we permute columns and take submatrices (delete rows and columns) to obtain $\tilde{\mathcal{V}}^{(r_1, \dots, r_k), \alpha([k])}$, transformations required by the Clifford and Clifford sampling algorithms. *A priori*, we need a new tree decomposition for each submatrix or column permutation, and these are no longer easy to find. However, in what follows, we present a systematic approach to obtain all tree decompositions of the required submatrices from the initial tree decomposition of $\mathcal{U}$ by performing a set of simple and efficient manipulations.

1. Permutation of columns

An important step is to reorder the (first n) columns of the matrix $\mathcal{U}$ by a permutation α . This acts as a graph isomorphism on $G(\mathcal{U})$:

$$|i\rangle \mapsto |\alpha^{-1}(i)\rangle. \quad (14)$$

This relabels columns without changing the connectivity structure, so we implement it on a tree decomposition again by just relabeling columns as in (14). The rest of the tree structure stays the same. Note that we conventionally act by α^{-1} within the column label, in order to address the matrix elements as $\mathcal{U}_{i, \alpha(j)}$. Formal details can be found in Appendix E 1.

2. Submatrices

In all of the Clifford and Clifford algorithms, we abundantly need to compute permanents of various submatrices. We need a way to exploit the known tree decomposition of the original matrix.

For any tree decomposition T of the matrix $\mathcal{U}$, and subsets $\mathcal{R}' \subseteq \mathcal{R}$ of rows and $\mathcal{C}' \subseteq \mathcal{C}$ of columns, selecting a submatrix $\mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$, we construct the restriction of T , denoted $T|_{\mathcal{R}', \mathcal{C}'}$, as the tree decomposition built on the same tree, where we simply delete all row and column labels not present in $\mathcal{R}'$ and $\mathcal{C}'$. We show an example in Figs. 8(a) and 8(b). As seen in Appendix E 2, this is a well-defined tree decomposition that indeed corresponds to the submatrix $\mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$. Importantly, this

cannot increase the treewidth, bounded (in Lemma E 4) as

$$\text{tw}(T|_{\mathcal{R}', \mathcal{C}'}) \leq \min(\text{tw}(T), |\mathcal{R}'| + |\mathcal{C}'| - 1). \quad (15)$$

3. Redundancy of nodes

Each node of our tree decomposition T^C contains a single column label. When restricting the decomposition, we may end up with a node $t \in T$ without any columns, i.e., $\kappa'(t) = \emptyset$. Indeed, our algorithm will perform such restrictions extensively. In this case, this node contains no information (corresponds to a column no longer present), and we can remove it, obtaining a new decomposition $T \setminus t$ where we directly connect the neighbors of t (unless it is root or leaf), see Figs. 8(c) and 8(d). We expand on the formal details in Appendix E 3.

V. ALGORITHM FOR SHALLOW CIRCUITS

Given the above methods, we could drop-in replace all permanent computation in CC-C by Algorithm CP to obtain a running time of $O(mn^2 2^\omega \omega^2)$. A similar method was used before in Ref. [11]. In what follows, however, we show how to achieve $O(n^2 2^\omega \omega^2) + O(\omega n^3)$ by finding an efficient way of computing the $k + 1$ permanents of submatrices $\mathcal{W}_{\diamond j}$ from (7b) using a single tree decomposition, leading to a scaling linear in k , reproducing the saving obtained in the original proposal of Clifford and Clifford [6]. The key tool is to compute the $k + 1$ permanents of submatrices $\mathcal{W}_{\diamond j}$ in the right order by doing local updates to the single initial (restricted) tree decomposition. This will allow us to exploit precomputed tables, reducing the amount of required computation at each step, closely mimicking the idea of CC-C. Note that, in order to reuse a global tree decomposition and tables computed on it, we require the noncollision setting of $m = \Omega(n^2)$.³

A. Preparation

We first prepare the main tree decomposition $T = T^C$ of the graph $G(\mathcal{U})$ of the banded unitary $\mathcal{U}$. Recall that each tree node contains a single column label (input of the circuit) and all row labels (outputs of the circuit) for which that column may have a nonzero element in $\mathcal{U}$; see Sec. IV B.

We have a fixed pattern of input photons to the circuit $|n\rangle$ with $n_i \in \{0, 1\}$ for each input mode i . We restrict the tree decomposition T by removing the columns corresponding to unoccupied input modes, as well as the corresponding tree nodes; see Secs. IV C 2 and IV C 3. We also remove those rows that only contain zeros after the deletion of columns. The new decomposition $\tilde{T}$ corresponds to the submatrix of $\mathcal{U}$, denoted $\mathcal{V}^n$, restricted to the columns corresponding to occupied inputs. The matrix $\mathcal{V}^n$ may apparently have a different band structure, but this is not a problem: the original structure is remembered by the restricted tree-decomposition whose treewidth remains at most $\omega = \text{tw}T$ by (15).

³This is because a collision leads to copies of an output vertex in $G(\tilde{\mathcal{V}}^{(r_1, \dots, r_k), \alpha([k])})$, along with copies of edges. Generally, this introduces new cycles, invalidating the global tree decomposition.

1. Permutation of inputs

Next, we generate a uniformly random permutation $\alpha \in S_n$ that we use to permute the columns by relabeling $|i\rangle$ to $|\alpha^{-1}(i)\rangle$ in $\tilde{T}$ as shown in (14). This preserves all tree decomposition structures while ensuring that the new decomposition correctly describes the permuted matrix. Note that after the permutation, the column label $|i\rangle$ is found in node $t_{\alpha(i)}$ of the tree.

2. Precomputing Q tables

We compute the Q tables of the entire tree for later use. As each table $Q[t]$ is local to the node t , each containing a single column, this consists of permanents of 1×1 matrices, i.e., copies of components of $\mathcal{V}^n$.⁴

B. The sampling algorithm

Now we perform the sampling itself. We show first the special case of the first marginal and then generalize, showing that not only can we efficiently compute the permanents needed for the Laplace expansion, but by doing our tree manipulations in the right order, we achieve the ability to reuse almost all dynamic programming tables for several calculations.

1. First marginal

The first marginal can be seen as expanding the empty sample to $k = 1$ photon. The Laplace expansion is trivial as the matrix $\mathcal{W}_{\diamond 1}$ is empty. Equivalently, we compute the permanents of 1×1 matrices. The probability of the first photon being in output mode i is $p_1(i) = |\mathcal{W}_{1,1}^{(i)}|^2$, where the subscript on p_1 indicates that this is the first marginal. This probability is the modulus squared of

$$\mathcal{W}_{1,1}^{(i)} = \tilde{\mathcal{V}}_{1,1}^{(i), (\alpha(1))} = \mathcal{U}_{i, \alpha(1)}, \quad (16)$$

where we note the use of notation referring to the original matrices (without column permutation), and write α explicitly for clarity. The matrix element $\mathcal{U}_{i, \alpha(1)}$ is represented by the node $t_{\alpha(1)}$ containing the label $|1\rangle$. We remark that one does not need to explore all m output modes, as at most ω [contained in $\rho(t_{\alpha(1)})$] are connected to the input mode $\alpha(1)$ due to the circuit depth. We select one output mode at random, using the computed marginal probability distribution p_1 , and store it in the register r_1 .

2. k th marginal

The k th marginal, for $k > 1$, adds the k th input photon to the sample of $k - 1$ distinct photon modes $(r_1, \dots, r_{k-1})$ which we have already obtained in previous steps. The latter determines the matrix $\mathcal{W} = \tilde{\mathcal{V}}^{(r_1, \dots, r_{k-1}), \alpha([k])}$ made of the restriction of $\mathcal{V}^n$ to rows $(r_1, \dots, r_{k-1})$, and to columns $\alpha([k])$. The k th marginal adds one more output photon, leading to $k \times k$ matrices $\mathcal{W}^{(r_k)}$, for all $r_k \in [m] \setminus \{r_1, \dots, r_{k-1}\}$. Recall

⁴We need these tables for consistency with the formulation of the Cifuentes and Parrilo algorithm, although in practice, one may choose to refer directly to the matrix.

that $\mathcal{W}^{(r_k)}$ is the matrix obtained from $\mathcal{W}$ by adding a row r_k of $\mathcal{V}^n$, restricted to the same columns.

The matrix $\mathcal{W}$ is represented by the restriction $T^{\mathcal{W}}$ of $\tilde{T}$ that removes all column labels other than $\langle 1 |, \dots, \langle k |$, which are found in tree nodes $t_{\alpha(1)}, \dots, t_{\alpha(k)}$, respectively. We also need to remove all output labels except $|r_1\rangle, \dots, |r_{k-1}\rangle$. Note that all nodes except for $t_{\alpha(1)}, \dots, t_{\alpha(k)}$ are redundant in this restriction, so we can skip them in the permanent computations. Equivalently, we delete the redundant nodes while keeping the remaining nodes connected, as in Fig. 8(d); see Sec. IV C 3.

The (unnormalized) marginal probability distribution p_k of the k th photon has values $p_k(r_k) = |\text{per}\mathcal{W}^{(r_k)}|^2$. We compute $\text{per}\mathcal{W}^{(r_k)}$ using the Laplace expansion from Eq. (7b):

$$\text{per}\mathcal{W}^{(r_k)} = \sum_{j=1}^k \mathcal{U}_{r_k, \alpha(j)} \text{per}\mathcal{W}_{\diamond j}.$$

We compute the collection $\{\text{per}\mathcal{W}_{\diamond j}\}_{j=1}^k$ of permanents of the shared submatrices using a collection of closely related tree restrictions $\{T_{\diamond j}^{\mathcal{W}}\}_{j=1}^k$. Each $T_{\diamond j}^{\mathcal{W}}$ is the restriction of $\tilde{T}$, the global decomposition of $\mathcal{V}^n$, to rows $\{r_1, \dots, r_{k-1}\}$ and columns $\alpha([k]) \setminus \{j\}$, that is, $T_{\diamond j}^{\mathcal{W}}$ is the tree decomposition representing the matrix $\mathcal{W}_{\diamond j}$.

3. Moving machine head: Local manipulations $T_{\diamond j}^{\mathcal{W}}$

We could already compute the Laplace expansion using $\{T_{\diamond j}^{\mathcal{W}}\}_{j=1}^k$, evaluating them in some arbitrary order. However, if we view this as a sequence of local manipulations to the overarching tree decomposition $T^{\mathcal{W}}$ corresponding to $\mathcal{W}$, and if we do these in an appropriate order, we unlock a further advantage of reusing the dynamic programming tables between the individual permanents.

The idea, as shown in an example in Fig. 9, where each subfigure corresponds to one step, is to imagine a machine head walking sequentially along the nodes of $T^{\mathcal{W}}$ from one end to the other, in analogy to a Turing machine walking the tape. At each step, it temporarily replaces the node where the head is, say t , by a dummy node d that contains no new information and sets this dummy d to be the root of the tree. Then it computes the permanent from this tree using the algorithm CP. Finally, it restores the original node t and moves on to the next one, repeating the procedure. We elaborate on the details in three conceptual parts:

a. Order of manipulations. Let $(j_1, \dots, j_k)$ be some permutation of $[k]$ that determines the order in which we compute the permanents, i.e., we start with $\text{per}\mathcal{W}_{\diamond j_1}$, continue with $\text{per}\mathcal{W}_{\diamond j_2}$, etc. To minimize the set of P tables changing between steps, choose the values $j_1, \dots, j_k$ so that we manipulate the nodes $t_{\alpha(j_1)}, \dots, t_{\alpha(j_k)}$ in the order as they appear in the tree of $T^{\mathcal{W}}$, i.e., $t_{\alpha(j_\ell)} = t_\ell$. Thus $j_\ell = \alpha^{-1}(\ell)$ for all ℓ .⁵ Note that, to simplify notation in what follows and in Fig. 9, we number the nodes of $T^{\mathcal{W}}$ consecutively as $t_1, t_2, \dots, t_k$, ignoring deleted nodes from $\tilde{T}$ where the numbering (but not the order!) may differ.

⁵We may also choose to walk the tree in the opposite direction. This does not make a difference.

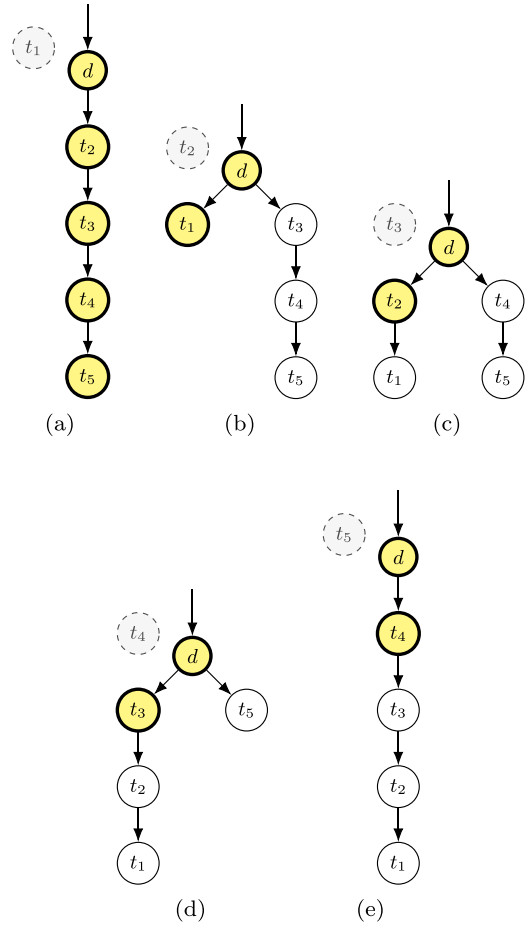


FIG. 9. Computation of $\text{per}\mathcal{W}^{(rs)}$ using the Laplace expansion: (a) We start with $T_{\diamond 1}^{\mathcal{W}}$ corresponding to $\text{per}\mathcal{W}_{\diamond \alpha^{-1}(1)}$. To compute this, replace the node t_1 by a dummy d and run algorithm CP. The P and Q tables of all nodes need to be computed, which is shown by yellow filling and a bold outline. (b) Next, put t_1 back and replace t_2 by a new dummy d that is now the parent of both t_1 , and of the subtree of t_3 , obtaining $T_{\diamond 2}^{\mathcal{W}}$, which is needed to compute $\text{per}\mathcal{W}_{\diamond \alpha^{-1}(2)}$. All $Q[t_1], \dots, Q[t_5]$ tables have been computed and remembered in step (a), so we do not recompute them. Observe that each of $t_3, \dots, t_5$ has the same child (if any) as in step (a), so their P tables do not need to be recomputed either. Marked in yellow are t_1 and d , the only nodes for which we need to compute new tables: only $P[t_1]$ for t_1 , and both $Q[d]$ and $P[d]$ for d . (c)–(e) The following steps, corresponding to $j = 3, 4, 5$, respectively. In each, we mark in yellow the nodes for which we need to compute the dynamic programming tables. At step j , the tables computed are $P[t_{j-1}]$, $Q[d]$, and $P[d]$. The latter contains $\text{per}\mathcal{W}_{\diamond \alpha^{-1}(j)}$ in $P[d](\rho'(d), \emptyset)$. Observe, for example, in panel (d) that nodes t_1 and t_2 are not yellow: following the same idea as before, their children (and hence P tables) have not changed from the previous step (c).

b. Node replacement. To compute $\text{per}\mathcal{W}_{\diamond j}$, we need to remove the column $\langle j |$ and thus node $t_{\alpha(j)}$ of $T^{\mathcal{W}}$, obtaining $T_{\diamond j}^{\mathcal{W}}$. We need to consider the part of the tree to the left and right of $t_{\alpha(j)}$, so conceptually it is useful to create a dummy node containing no information instead of connecting the neighbors of $t_{\alpha(j)}$ directly. To satisfy Axiom (T3), $\rho'(d) = \rho'(t_a) \cap \rho'(t_b)$ if d has neighbors t_a and t_b , and $\rho'(d) = \emptyset$ if d is at either end of the tree, where ρ' is the row function of $T_{\diamond j}^{\mathcal{W}}$. It contains no

columns. After computing $\text{per}\mathcal{W}_{\diamond j}$, we return the node $t_{\alpha(j)}$ and move on to the next node with a new dummy.

From now on, it will be useful to refer to nodes by their order in the tree as t_j , instead of $t_{\alpha(j)}$ used above to more easily address matrix columns. Observe that t_j corresponds to column $\langle \alpha^{-1}(j) \rangle$, and removing it allows computing $\text{per}\mathcal{W}_{\diamond \alpha^{-1}(j)}$.

c. Setting the root. Finally, at each step, we make the new dummy d the root of the tree by orienting edges away from it. This means the permanents computed will be found in $P[d](\rho'(d), \emptyset)$. Together with the correct order of permanents, this allows us to reuse previously computed P tables in subsequent steps. We show the idea in Fig. 9, where we highlight the nodes for which we need to compute new P tables. Recall that Q tables stay the same (except for the new dummy node).

Using the example, observe that it is only in the first step [$j = 1$, Fig. 9(a)] where we need to compute all of the P tables. Going to step $j = 2$ [Fig. 9(b)], neither the node t_3 nor its subtree T_{t_3} changed, and their P tables stay the same. We must (re)compute only the tables $P[t_1]$ whose subtree T_{t_1} changed, and $Q[d]$ along with $P[d]$, corresponding to a new node. Observe that when going to step $j = 3$ [Fig. 9(c)], the contents and subtree of t_1 stay the same. This generalizes to the left subtree $t_1, \dots, t_{j-2}$ staying the same for every step $j > 2$, requiring no new table computation. We thus conclude that in every step $j > 1$, we need only compute at most three new tables.⁶

4. Postprocessing

Finally, having generated an outcome $\underline{r} = (r_1, \dots, r_n)$ of size n , we follow the Clifford and Clifford [6] algorithms: forget the identity of photons, and obtain the sample $\underline{n}'$ (vector of mode occupations) by counting how many times each outcome mode appears in $\underline{r}$.

C. Running time asymptotic analysis

To compute the running time, we note first that the computation of a Q table takes time $O(2^\omega \omega^2)$, and that of a P table of a node with c children takes time $O(c 2^\omega \omega^2)$ [10]. We derive these results in Appendix C2, specifically Lemmas C2 and C3, respectively. Both of these bounds depend on the size of the contents of the corresponding tree node, which is bounded by the treewidth ω of T^C . Even though we work ubiquitously with restrictions that may decrease treewidth, we still keep this general bound on the treewidth for simplicity.

In the following, we focus on the sampling of marginals $k = 2, \dots, n$, which subsumes the preparation, sampling the first marginal ($k = 1$), as well as the postprocessing.

To prepare for the Laplace expansion (7b) with a given output r_k , we iterate through the tree as described in the Sec. VB3, computing k permanents. For the first step $j = 1$, we compute the P tables for all k nodes that have each at most one child, taking time $O(k 2^\omega \omega^2)$. Then at each step $j > 1$, we

only compute up to three new dynamic programming tables $P[t_{j-1}]$, $Q[d]$, and $P[d]$, where t_{j-1} has at most one child, and d has at most two. Step $j > 1$ thus takes time $O(2^\omega \omega^2)$. In total, the computation of the k permanents needed in Laplace expansion takes the time of $O(k 2^\omega \omega^2)$.

Sampling r_k takes time $O(k^2 \omega)$ because the Laplace expansion needs k scalar multiplications per possible output, and there are at most $k\omega$ nonzero outputs. Here, we recall that each column has at most ω rows with nonzero elements, and at this k th step, we have access to k of those. Note also that this is an upper bound: some columns may be overlapping, in which case the number of possible values of r_k will be lower.

The loop that adds a new photon runs for k in the range from 2 to n . Putting this all together, generating a single sample using our algorithm needs the following asymptotic running time:

$$\sum_{k=2}^n O(k 2^\omega \omega^2) + O(k^2 \omega) = O(n^2 2^\omega \omega^2) + O(\omega n^3).$$

Remark that $O(\omega n^3)$ remains better than $O(mn^2)$ while $\omega n < m$. If $\omega n > m$ it is easy to see that, by the design of our algorithm, we recover the scaling $O(mn^2)$.

For completeness, we shortly remark that the tree manipulations, i.e., column permutations and restrictions of tree decompositions, which consist of computing set intersections, are subsumed by the above complexity.

VI. CONCLUSION AND FUTURE WORK

We have presented an algorithm that generates samples of n photons from a boson sampling experiment on a shallow circuit in the no-collision regime $m = \Omega(n^2)$, a necessary constraint in our algorithm. Combining the central idea of the boson sampling simulation work of Clifford and Clifford and the computation of the permanent of matrices with fixed treewidth of Cifuentes and Parrilo, we have constructed an algorithm that generates a sample of n photons from a boson sampling experiment in an m -mode circuit described by a matrix of treewidth ω that runs in time $O(n^2 2^\omega \omega^2) + O(\omega n^3)$. For the case of a shallow circuit, with depth $D = c \log n$, the sampling algorithm becomes polynomial time $O(n^{2(c+1)} \log^2 n) + O(n^3 \log n)$, as suggested in Ref. [9].

A straightforward use of the Cifuentes and Parrilo result leads to a scaling of $O(mn^2 2^\omega \omega^2)$ or $O(n^3 2^\omega \omega^2)$. The former result was attained in Ref. [11] by replacing the permanent computations by the algorithm CP, but without optimizing the Laplace expansion on the tree decomposition. Our key technical improvement is to restore the idea of Clifford and Clifford to decouple the dominant running time term from the number of output modes m using the Laplace expansion and precomputing submatrices $\mathcal{W}_{\diamond j}$ in the same running time as needed to compute a permanent of size k . We adapt this idea to the tree decomposition framework of Cifuentes and Parrilo by performing a series of local updates to single nodes that allows us to compute k permanents of subtrees in time comparable to the computation of the permanent of the full tree.

In the regime $\omega = O(n)$ our sampling algorithm has an extra polynomial cost with respect to the traditional

⁶We need only compute one table if, in each step j , instead of adding a dummy d , we only remove t_j and root the tree at t_{j-1} .

Clifford and Clifford algorithm. An interesting open question would be to find an algorithm that has better scaling than ours and that converges to the exact running time $O(n2^n)$ when $\omega = n$. Another interesting open problem would be to generalize our result to the collision scenario, where multiple photons can be measured in the same mode. On face value, the latter complicates the use of tree decompositions, because such collisions cause the copying of rows of the matrix whose permanent represents the amplitude, creating a new cycle in its graph. Thus, we expect that one would likely need to construct entirely new tree decompositions in such cases.

Finally, it would be interesting to see whether our ideas can be adapted to circuits that allow nonlocal operations, such as the scheme of Ref. [21]. These can obtain a dense unitary with only logarithmic circuit depth, posing a seemingly difficult challenge for tree-decomposition-based methods that rely on matrix sparsity. Perhaps if one were able to reintroduce sparsity, for example by exploiting loss to truncate small matrix entries, one may be able to find a good tree decomposition. In such a case, it should be possible to either adapt our method or perhaps use our work as a toolkit to develop a different approach to tackle this problem.

ACKNOWLEDGMENTS

This work is partially based on work done by S.N. as his 2022 undergraduate final project [22] supervised by R.G.-P., and continued as an LFCS internship over the summer 2022. R.G.-P. was supported by the EPSRC-funded project Benchmarking Quantum Advantage. S.N. thanks William Clements for helpful discussions.

DATA AVAILABILITY

No data were created or analyzed in this study.

APPENDIX A: BOSON SAMPLING

Here, we expand the details on quantum optics and boson sampling omitted in Sec. II. Recall that in an ideal boson sampling circuit, the input and output can be written in a basis of quantum states corresponding to the location of n photons in m modes, represented as number (Fock) states $|\underline{n}\rangle = |n_1, \dots, n_m\rangle \in (\mathbb{C}^n)^{\otimes m}$, where $n_i \in \mathbb{N}$ is the number of photons in mode i , and $\sum_i n_i = n$. These states are canonically written using the creation operators as

$$|\underline{n}\rangle = \prod_{i=1}^m \frac{(\hat{a}_i^\dagger)^{n_i}}{\sqrt{n_i!}} |0\rangle^{\otimes m},$$

where $\hat{a}_i^\dagger$ acts only on mode i . An interferometer is a linear optical device that is represented by a unitary matrix $\mathcal{U} \in \mathbb{U}(m)$ acting linearly on the m creation operators as

$$\hat{a}_i^\dagger \xrightarrow{\mathcal{U}} \sum_{j=1}^m \mathcal{U}_{i,j} \hat{a}_j^\dagger \quad \text{for all } i = 1, \dots, m. \quad (\text{A1})$$

The unitarity of $\mathcal{U}$ (i.e., $\mathcal{U}^\dagger \mathcal{U} = \mathcal{U} \mathcal{U}^\dagger = \mathbb{1}$) ensures the total number of photons is preserved [14].

1. First-quantization representation

We now reformulate boson sampling in first quantization, used in Sec. II A. For a given occupation state $|\underline{n}\rangle$, we start by writing an n -qudit state where each m -dimensional qudit represents a photon in a specified mode, and these are arranged in a nondecreasing order:

$$|\underline{z}\rangle := |1\rangle^{\otimes n_1} \otimes |2\rangle^{\otimes n_2} \otimes \dots \otimes |m\rangle^{\otimes n_m}. \quad (\text{A2})$$

Any permutation σ of the n qudits will provide a new basis state $|\underline{r}\rangle = \sigma |\underline{z}\rangle$ that has the same mode occupation pattern as $|\underline{z}\rangle$, given by $|\underline{n}\rangle$. The states corresponding to the same number state are considered equivalent, written $|\underline{z}\rangle \sim |\underline{r}\rangle$. The size of each equivalence class of states $|\underline{r}\rangle$ with the same photon distribution as $|\underline{z}\rangle$ is given by

$$\binom{n}{n_1, \dots, n_m} = \frac{n!}{\prod_{i=1}^m n_i!}. \quad (\text{A3})$$

In the notation of $|\underline{z}\rangle$ and $|\underline{r}\rangle$, the qudits are assigned specific locations and are therefore distinguishable. To ensure indistinguishability, the state $|\Phi_n\rangle$ corresponds to the symmetric superposition of all their permutations:

$$|\Phi_n\rangle = \frac{1}{\sqrt{n! \prod_{i=1}^m n_i!}} \sum_{\sigma \in S_n} \sigma |\underline{z}\rangle, \quad (\text{A4})$$

where σ is a permutation in the symmetric group on n elements S_n , and the prefactor on the RHS is its normalization [13]. Note this is Eq. (2).

The action of the interferometer $\mathcal{U}$ in the first-quantization representation follows from (A1). For a single qudit $|i\rangle$, the action reads

$$|i\rangle \xrightarrow{\mathcal{U}} \sum_{j=1}^m \mathcal{U}_{i,j} |j\rangle = \mathcal{U} |i\rangle, \quad (\text{A5})$$

which results from its action on $\hat{a}_i^\dagger$ and the fact that $|i\rangle = \hat{a}_i^\dagger |0\rangle$. Likewise, product states correspond to products of creation operators, which give the relation

$$|\underline{z}\rangle \equiv \bigotimes_{k=1}^n |z_k\rangle \xrightarrow{\mathcal{U}} \bigotimes_{k=1}^n (\mathcal{U} |z_k\rangle) = \mathcal{U}^{\otimes n} |\underline{z}\rangle, \quad (\text{A6})$$

where we have introduced the notation $|z_j\rangle$ to mean the j th qudit in the product (A2). By linearity, we obtain that $|\Phi_n\rangle \mapsto \mathcal{U}^{\otimes n} |\Phi_n\rangle$, i.e., in the first-quantization representation, the linear optical circuit unitary over n photons acts as $\mathcal{U}^{\otimes n}$.

2. Boson sampling outcome probabilities

We fill in the derivation behind equation (6a), which follows the steps in Ref. [13]. We represent $|\underline{n}\rangle \cong |\Phi_n\rangle$ canonically by $|\underline{z}\rangle$ defined in (A2), we analogously represent the output state $|\underline{n}'\rangle$ in qudit space by $|\underline{z}'\rangle$, again with tensor factors ordered in nondecreasing order. We denote by $|z_j\rangle$ the j th photon (tensor factor) of $|\underline{z}\rangle$, and we define $|z'_j\rangle$ for $|\underline{z}'\rangle$ analogously.

Recall from (A6) that in the qudit representation, the interferometer $\mathcal{U}$ acts as the n -fold tensor product $\mathcal{U}^{\otimes n}$. Then the probability of measuring the output state $|\underline{n}'\rangle$ is given by the Born rule $\mathbb{P}[\underline{n}'|\underline{n}] = |\langle \underline{n}' | \mathcal{U} | \underline{n} \rangle|^2$. We then write this in the

qudit representation:

$$\begin{aligned}
 \mathbb{P}[\underline{n}'|\underline{n}] &= |\langle \Phi_{\underline{n}'} | \mathcal{U}^{\otimes n} | \Phi_{\underline{n}} \rangle|^2 \\
 &\stackrel{(A4)}{=} \left| \left(\frac{1}{\sqrt{n! \prod_{i=1}^m n_i!}} \sum_{\sigma \in S_n} \langle \underline{z}' | \sigma^{-1} \rangle \right) \mathcal{U}^{\otimes n} \right. \\
 &\quad \times \left. \left(\frac{1}{\sqrt{n! \prod_{i=1}^m n_i!}} \sum_{\tau \in S_n} \tau | \underline{z} \rangle \right) \right|^2 \\
 &= \left| \frac{1}{n! \sqrt{\prod_{i=1}^m n_i! n_i!}} \sum_{\sigma, \tau \in S_n} \left(\bigotimes_{j=1}^n \langle z'_j | \right) \right. \\
 &\quad \times \left. \sigma^{-1} \mathcal{U}^{\otimes n} \tau \left(\bigotimes_{k=1}^n | z_k \rangle \right) \right|^2,
 \end{aligned}$$

where we expanded the definition of $|\underline{n}\rangle$, $|\underline{n}'\rangle$, rearranged the sums over the permutations σ and τ , and decomposed $|\underline{z}\rangle$, $|\underline{z}'\rangle$ into their individual qudits. Note that the action of the permutations is by braiding, rearranging tensor factors; this means that the adjoint corresponds to the inverse: $\sigma^\dagger = \sigma^{-1}$, as in $\langle \Phi_{\underline{n}'} |$ above.

The permutation σ^{-1} , acting as a braiding of the tensor factors, commutes with the operator $\mathcal{U}^{\otimes}$, so we can write

$$\begin{aligned}
 \mathbb{P}[\underline{n}'|\underline{n}] &= \left| \frac{1}{n! \sqrt{\prod_{i=1}^m n_i! n_i!}} \sum_{\sigma, \tau \in S_n} \left(\bigotimes_{j=1}^n \langle z'_j | \right) \right. \\
 &\quad \times \left. \mathcal{U}^{\otimes n} \underbrace{\sigma^{-1} \tau}_{\zeta} \left(\bigotimes_{k=1}^n | z_k \rangle \right) \right|^2 \\
 &= \left| \frac{n!}{n! \sqrt{\prod_{i=1}^m n_i! n_i!}} \sum_{\zeta \in S_n} \left(\bigotimes_{j=1}^n \langle z'_j | \right) \right. \\
 &\quad \times \left. \mathcal{U}^{\otimes n} \left(\bigotimes_{k=1}^n | z_{\zeta^{-1}(k)} \rangle \right) \right|^2,
 \end{aligned}$$

where we define the composite $\zeta = \sigma^{-1} \tau$ in S_n . The composite ranges over the whole group S_n , so we remove one summation and get a factor of $n!$ coming from the standard group-theoretic fact that every ζ is a composite of $n!$ different pairs of σ and τ . Photons in the product $\bigotimes_k | z_k \rangle$ are reordered by ζ : the k th photon is sent to $\zeta(k)$. We rewrite this state with the permutation inside. To find which is the k th factor after the permutation, we apply the inverse ζ^{-1} inside the state to obtain $\bigotimes_k | z_{\zeta^{-1}(k)} \rangle$.

Finally, we match up the tensor factors across the expression, giving us terms with $j = \zeta^{-1}(k)$. This reduces to a product of bra-kets with $\mathcal{U}$ in the middle. To bring the expression into a form of a permanent, we replace the permutation ζ by its inverse $\pi = \zeta^{-1}$. This reorders the summands, but the summation commutes, so the value is the same. We obtain

$$\begin{aligned}
 \mathbb{P}[\underline{n}'|\underline{n}] &= \left| \frac{1}{\sqrt{\prod_{i=1}^m n_i! n_i!}} \sum_{\pi \in S_n} \prod_{j=1}^n \mathcal{V}_{j, \pi(j)}^{\underline{n}', \underline{n}} \right|^2 \\
 &= \frac{|\text{per} \mathcal{V}^{\underline{n}', \underline{n}}|^2}{\prod_{i=1}^m n_i! n_i!}.
 \end{aligned}$$

These are Eqs. (6a) and (6b) in Sec. II A 1. Recall that we define the matrix $\mathcal{V}^{\underline{n}', \underline{n}}$ with components

$$\mathcal{V}_{i,h}^{\underline{n}', \underline{n}} := \langle z'_i | \mathcal{U} | z_h \rangle,$$

i.e., the matrix elements of $\mathcal{U}$ at row (output) z'_i and column (input) z_h .

APPENDIX B: CLIFFORD AND CLIFFORD ALGORITHMS

In Sec. II B 1, we shortly sketched the main ideas of algorithms CC-A and CC-B. We now elaborate, showing the conceptual steps required to construct these algorithms.

1. Variable origins of photons and algorithm CC-A

The main idea of CC-A is to perform the sampling in the $|\underline{r}\rangle$ basis and grow the sample photon-by-photon, using a standard chain rule procedure. The chain rule states

$$p(r_1, \dots, r_n) = p(r_1) p(r_2 | r_1) \cdots p(r_n | r_{n-1}, \dots, r_1). \quad (\text{B1})$$

If one has access to all marginals of $p(r_1, r_2, \dots, r_n)$, one can generate a sample using the chain rule by first computing $p(r_1)$ for all $r_1 \in [m] = \{1, \dots, m\}$, and committing to a choice of r_1 following the computed probability distribution. Having fixed r_1 , we compute $p(r_1, r_2)$ for all $r_2 \in [m]$, using $p(r_2 | r_1) = p(r_1, r_2) / p(r_1)$, and again commit to a choice of r_2 following that distribution. Here, we use the value $p(r_1)$ computed in the previous step. Iterating this procedure, we can generate one sample via the computation of mn marginals.

a. Marginal probabilities

We follow the steps in Ref. [6] to derive the marginal probabilities. Equation (6b) gives the probability of outcome $|\underline{n}'\rangle$ for an arbitrary input $|\underline{n}\rangle$. We follow the convention that input modes are occupied by up to one photon ($n_i \leq 1$), and, without loss of generality, use the standard input $|\underline{n}\rangle = |1\rangle^{\otimes n} \otimes |0\rangle^{\otimes(m-n)}$, i.e., the photons are found in the first n modes. This allows us to simplify notation in the following, omitting the reference to $|\underline{n}\rangle$:

$$\mathbb{P}[\underline{n}'] = \frac{|\text{per} \mathcal{V}^{\underline{n}', (1^n, 0^{m-n})}|^2}{\prod_{i=1}^m n_i!}.$$

Define $\tilde{\mathcal{V}}_{i,j}^{(r_1, \dots, r_k), C}$ which fulfils the same function as $\mathcal{V}^{\underline{n}', \underline{n}}$, but it is given by listing which modes appear in the qudit representation and allows partial samples. Its components are

$$\tilde{\mathcal{V}}_{i,j}^{(r_1, \dots, r_k), C} = \langle r_i | \mathcal{U} | c_j \rangle,$$

where the input qudits are given as an (unordered) set $C = \{c_1, \dots, c_k\}$, so we impose $c_1 \leq c_2 \leq \dots \leq c_k$ for convenience. We can choose any ordering here, because the permanent is invariant under permutations of columns, as well as rows. Note that we always give the output photons $(r_1, \dots, r_k)$ in the order in which they are sampled. The latter ensures that the newest photon, for which we compute the marginal probability later, corresponds to the last row of the matrix. Note further that only the presence or absence of a row or column is relevant (or, more generally, its number of repetitions), but we choose an ordering so that we can write down concrete matrices. The set C is the set of input modes,

i.e., columns of $\mathcal{U}$, and the (full) output state is represented by $|\underline{z}'\rangle = |z'_1\rangle \otimes \cdots \otimes |z'_n\rangle$. We write the probability in this notation, and define the pmf q :

$$q(\underline{z}') := \frac{|\text{per} \tilde{\mathcal{V}}^{(z'_1, \dots, z'_n), [n]}|^2}{\prod_{i=1}^m n'_i!} = \mathbb{P}[\underline{n}'], \quad (\text{B2})$$

where we recall the notation $[n] = \{1, \dots, n\}$.

By rewriting the probability in terms of qudits, we can define a new pmf where the photon samples are given in the $|\underline{r}\rangle$ basis. That is, we now work with distinguishable photons, specifying which photon from the input ends up in which output mode. We write the new pmf p defined on all $\underline{r} \in [m]^n$:

$$p(\underline{r}) := \mathbb{P}[\underline{r}] = \frac{|\text{per} \tilde{\mathcal{V}}^{[\underline{r}], [n]}|^2}{n!}. \quad (\text{B3})$$

The permanent is invariant under permutation of rows and columns, so $p(\underline{r}) = p(\underline{z}')$, where $\underline{z}'$ are the same modes in nondecreasing order, representing the same number state, i.e., $|\underline{r}\rangle \sim |\underline{z}'\rangle$. We recall from (A3) that there are $n!/\prod_{i=1}^m n'_i!$ such vectors $\underline{r}$. The two pmfs are related as

$$\begin{aligned} \sum_{\underline{r} \sim \underline{z}'} p(\underline{r}) &= p(\underline{z}') \sum_{\underline{r} \sim \underline{z}'} 1 \\ &\stackrel{(\text{A3})}{=} \frac{n!}{\prod_{i=1}^m n'_i!} \times p(\underline{z}') \\ &\stackrel{(\text{B3})}{=} \frac{n!}{\prod_{i=1}^m n'_i!} \times \frac{|\text{per} \tilde{\mathcal{V}}^{[\underline{z}'], [n]}|^2}{n!} \\ &\stackrel{(\text{B2})}{=} q(\underline{z}'), \end{aligned}$$

which is the result we expect, and we conclude

$$p(\underline{r}) = p(\underline{z}') = \frac{\prod_{i=1}^m n'_i!}{n!} \cdot q(\underline{z}'). \quad (\text{B4})$$

This allows us to compute p instead of q .

Using the qudit representation and working in the $|\underline{r}\rangle$ basis, accessible thanks to the pmf p , allows us to work with partial samples of $k < n$ photons. However, the definition of $\mathcal{V}^{(z'_1, \dots, z'_k), C}$ requires us to select which of the incoming photons were measured in the partial sample. The probability of measuring some partial sample $\underline{r}^{(k)} := (r_1, \dots, r_k) \in [m]^k$ of k photons has to sum over the $\binom{n}{k} = n!/[k!(n-k)!]$ possible subsets of input modes from which these photons came. We arrive at the following:

Lemma B.1 (partial sample). The marginal pmf of a partial sample of $k \in [n]$ photons is

$$p(\underline{r}^{(k)}) = \frac{(n-k)!}{n!} \sum_{\substack{C \subseteq [n] \\ |C|=k}} |\text{per} \tilde{\mathcal{V}}^{(\underline{r}^{(k)}, C)}|^2. \quad (\text{B5})$$

Proof. To get the marginal pmf, we sum the full pmf over the part of sample $\underline{r}$ that is not in $\underline{r}^{(k)}$, which we call $\underline{r}' = (r_{k+1}, \dots, r_n)$:

$$\begin{aligned} p(\underline{r}^{(k)}) &= \sum_{\underline{r}'} p(\underline{r}) \stackrel{(\text{B3})}{=} \sum_{\underline{r}'} \frac{|\text{per} \tilde{\mathcal{V}}^{[\underline{r}], [n]}|^2}{n!} \\ &= \frac{1}{n!} \sum_{\underline{r}'} \left(\sum_{\sigma \in S_n} \prod_{j=1}^n \mathcal{U}_{r_j, \sigma_j} \right) \left(\sum_{\tau \in S_n} \prod_{\ell=1}^n \mathcal{U}_{r_\ell, \tau_\ell} \right)^*, \end{aligned}$$

where we expanded the definition of permanent and expressed the magnitude squared $|x|^2 = xx^*$ for any $x \in \mathbb{C}$. Recall that $\tilde{\mathcal{V}}^{[\underline{r}], [n]}$ is the matrix with rows $r_1, \dots, r_n$ and columns $[n]$ of $\mathcal{U}$. Below, we write $\sigma_j := \sigma(j)$, likewise for τ , to shorten the notation:

$$\begin{aligned} p(\underline{r}^{(k)}) &= \frac{1}{n!} \sum_{\underline{r}'} \sum_{\sigma, \tau \in S_n} \prod_{j=1}^n \mathcal{U}_{r_j, \sigma_j} \mathcal{U}_{r_j, \tau_j}^* \\ &= \frac{1}{n!} \sum_{\underline{r}'} \sum_{\sigma, \tau \in S_n} \prod_{j=1}^n \mathcal{U}_{\tau_j, r_j}^\dagger \mathcal{U}_{r_j, \sigma_j}. \end{aligned} \quad (\text{B6})$$

We have rearranged the expressions. In (B6), we used the fact that $\mathcal{U}_{r_j, \tau_j}^* = \mathcal{U}_{\tau_j, r_j}^\dagger$. We move the terms not summed over outside of the summation over $\underline{r}' = (r_{k+1}, \dots, r_n)$:

$$\begin{aligned} p(\underline{r}^{(k)}) &= \frac{1}{n!} \sum_{\sigma, \tau \in S_n} \left(\prod_{j=1}^k \mathcal{U}_{\tau_j, r_j}^\dagger \mathcal{U}_{r_j, \sigma_j} \right) \sum_{\underline{r}'} \prod_{\ell=k+1}^n \mathcal{U}_{r_\ell, \tau_\ell}^\dagger \mathcal{U}_{r_\ell, \sigma_\ell} \\ &= \frac{1}{n!} \sum_{\sigma, \tau \in S_n} \left(\prod_{j=1}^k \mathcal{U}_{\tau_j, r_j}^\dagger \mathcal{U}_{r_j, \sigma_j} \right) \underbrace{\prod_{\ell=k+1}^n \sum_{r_\ell=1}^m \mathcal{U}_{r_\ell, \tau_\ell}^\dagger \mathcal{U}_{r_\ell, \sigma_\ell}}_{[\mathcal{U}^\dagger \mathcal{U}]_{\tau_\ell, \sigma_\ell}}. \end{aligned} \quad (\text{B7})$$

In (B7), we exchange the product and the sum by distributivity. Only one variable (r_ℓ) appears now, but the product ranges over ℓ , so we still have all the terms. Note that the summation over r_ℓ is a matrix multiplication $\mathcal{U}^\dagger \mathcal{U}$. As $\mathcal{U}$ is unitary, this product is the identity. The product over ℓ then becomes $\prod_{\ell=k+1}^n \delta_{\tau_\ell, \sigma_\ell}$, where $\delta_{i,j}$ is the Kronecker delta. The only nonzero terms are those where $\sigma_\ell = \tau_\ell$ for all $\ell > k$. We partition the permutations as $\sigma = \mu \oplus \zeta$ and $\tau = \nu \oplus \zeta$ into a shared part $\zeta \in S_{n-k}$ (this is the part that ensures $\sigma_\ell = \tau_\ell$ for $\ell > k$), and $\mu, \nu \in S_k$, which may be different. This means we also partition the set $[n]$ on which σ, τ act into a subset C of size k on which μ, ν act, and $[n] \setminus C$ acted upon by ζ .⁷

We rewrite the summation over σ and τ as summations over the selection of $C \subseteq [n]$, i.e., over the partitions, such that we select the part of $[n]$ where μ, ν act, and as summations over μ, ν themselves. For each C , there are $(n-k)!$ possible permutations ζ ; this gives a new factor in (B8):

$$\begin{aligned} p(\underline{r}^{(k)}) &= \frac{(n-k)!}{n!} \sum_{\substack{C \subseteq [n] \\ |C|=k}} \left(\sum_{\mu \in S_C} \prod_{j=1}^k \mathcal{U}_{r_j, \mu_j} \right) \left(\sum_{\nu \in S_C} \prod_{\ell=1}^k \mathcal{U}_{r_\ell, \nu_\ell} \right)^* \\ &= \frac{(n-k)!}{n!} \sum_{\substack{C \subseteq [n] \\ |C|=k}} |\text{per} \tilde{\mathcal{V}}^{(\underline{r}^{(k)}, C)}|^2. \end{aligned} \quad (\text{B8})$$

⁷By a partition $\sigma = \mu \oplus \zeta$, we mean that $\sigma_\ell = \mu_\ell$ for $i \leq k$ and $\sigma_\ell = \zeta_{\ell-k} + k$ for $\ell > k$. Note we abuse notation by saying $\sigma = \mu \oplus \zeta$ and $\tau = \nu \oplus \zeta$, hiding the detail that the output may need to be rearranged by some shared $\lambda \in S_n$ as it may not fit into the blocks $[k]$ and $[n-k]$. In reality, we have the composites $\sigma = \lambda(\mu \oplus \zeta)$ and $\tau = \lambda(\nu \oplus \zeta)$. However, these are formal details that would only obscure the meaning of the steps.

Thus, we have recovered the permanent of submatrices, and this concludes the proof. ■

b. Running time

The algorithm CC-A samples from $p(r_1, \dots, r_n)$ by committing to the partial sample $(r_1, \dots, r_{k-1})$ and extending it by a single r_k at each step $k = 2, \dots, n$. For every k we need to compute m marginals, each consisting of a sum of size $\binom{n}{k}$ and a permanent of a matrix of size $k \times k$ taking the time $O(k2^k)$ steps, each using the algorithm by Ryser [4] or Glynn [5]. This leads to a total running time:

$$O\left(m \sum_{k=1}^n k2^k \binom{n}{k}\right) = O(mn3^n). \quad (\text{B9})$$

At each step k , we need to remember the values $\{p(r_k | r_1, \dots, r_{k-1})\}_{r_k=1, \dots, m}$, i.e., store a vector of m values. Therefore, we require $O(m)$ additional memory, less than the memory needed to store the matrix $\mathcal{U}$ of size $O(m^2)$.

2. Removing the costly sum: Algorithm CC-B

The first improvement of CC-A suggested by Clifford and Clifford [6], which we refer to as CC-B, is to remove the costly sum over $\binom{n}{k}$ terms corresponding to sets C by preselecting them based on a random choice and only using the inputs (columns) in the chosen C . The idea is to remove the selection from the marginal itself and instead determine the order in which input photons appear. This order is chosen randomly for each full sample. This means each sample comes from a simpler probability distribution, but the ensemble of many samples follows the correct distribution.

As part of the initialization of the algorithm, i.e., before the sample is generated, choose a uniformly random permutation $\alpha \in S_n$. At each k , the set C will be defined to be $C = \alpha([k]) = \{\alpha(1), \dots, \alpha(k)\}$. For each C of size k , there are $k!(n-k)!$ permutations α such that $C = \alpha([k])$, so we need to divide by this in (B5) to preserve equality:

$$\begin{aligned} p(r^{(k)}) &= \frac{1}{n! k!} \sum_{\alpha \in S_n} |\text{per} \tilde{\mathcal{V}}_{r^{(k)}, \alpha([k])}^2| \\ &= \mathbb{E}_{\alpha \in S_n} \left[\underbrace{\frac{1}{k!} |\text{per} \tilde{\mathcal{V}}_{r^{(k)}, \alpha([k])}^2|}_{\phi(r^{(k)} | \alpha)} \right]. \end{aligned} \quad (\text{B10})$$

We see that the original pmf p is the expectation of a different pmf called ϕ , taken over the uniformly random $\alpha \in S_n$, where ϕ is conditioned on α . The expectation value over many samples, each with a new uniformly random α , leads to the correct marginal probability.

For each photon k , the $1/k!$ in (B10) is a shared constant prefactor for each value of r_k . Similarly to CC [6], for convenience, we can omit this shared prefactor in the algorithm and sample from unnormalized pmfs. This is possible using Walker's alias method in time $O(m)$ for a pmf stored in an

Algorithm CC-C: Boson Sampling

Input:

- number of modes m , and of photons n , $n \leq m$
- matrix $\mathcal{U} \in \mathbb{U}(m)$ representing the interferometer

Result:

 a single sample $\underline{z}'$

```

1  $\alpha \leftarrow$  uniformly random permutation from  $S_n$ ;
2  $\mathcal{V}^n \leftarrow$  first  $n$  columns of  $\mathcal{U}$  permuted using  $\alpha$ ;
3 initialize  $\underline{r} = (r_1, \dots, r_n) \leftarrow (0, \dots, 0)$ ;
4  $w_1(i) \leftarrow |\mathcal{V}_{i,1}^n|^2$  for all  $i \in [m]$ ; // first photon
5  $r_1 \leftarrow$  sample from the pmf  $w_1$ ;
6 for  $k \leftarrow 1, \dots, n-1$  do // rest of the sample
7    $\mathcal{W} \leftarrow$  rows  $r_1, \dots, r_k$  and first  $k+1$  columns of  $\mathcal{V}^n$ ;
   // corresponding to  $\tilde{\mathcal{V}}_{(r_1, \dots, r_k), \alpha([k+1])}$ 
8   compute  $\{\text{per } \mathcal{W}_{\diamond i}\}_{i=1}^{k+1}$ 
9    $w_{k+1}(i) \leftarrow \left| \sum_{j=1}^{k+1} \mathcal{V}_{i,j}^n \text{per } \mathcal{W}_{\diamond j} \right|^2$  for all  $i \in [m]$ 
10   $r_{k+1} \leftarrow$  sample from the (unnormalised) pmf  $w_{k+1}$ 
11  $\underline{z}' \leftarrow$  sort  $\underline{r}$  in non-decreasing order
12 return  $\underline{z}'$ , which represents  $|\Phi_{\underline{z}'}\rangle = \frac{1}{(\dots)} \sum_{\sigma \in S_n} \sigma |\underline{z}'\rangle$ 
```

FIG. 10. Pseudocode of Algorithm CC-C. Note the differing convention with respect to the main text where we expand from a sample of $k-1$ to k photons. In the present Appendix, for notational simplicity, we expand from k to $k+1$ photons.

array of length m , the latter being the number of possible new modes [23].⁸

a. Running time

The running time of Algorithm CC-B scales as

$$O\left(m \sum_{k=1}^n k2^k\right) = O\left(mn \sum_{k=1}^n 2^k\right) = O(mn2^{n+1}). \quad (\text{B11})$$

Hence CC-B generates a sample in time $O(mn2^n)$. Similarly to CC-A, it requires additional space $O(m)$, where m is the number of modes.

3. Laplace expansion: Algorithm CC-C

In Sec. II B 2, we describe the Laplace expansion giving the algorithm CC-C, the final improvement of Ref. [6]. For convenience, we give in Fig. 10 a pseudocode that summarizes the steps.

Running time

The running time of CC-C is

$$\sum_{k=1}^n O(k2^k) + O(mk) = O\left(n \sum_{k=1}^n 2^k\right) + O\left(m \sum_{k=1}^n k\right) \quad (\text{B12a})$$

$$= O(n2^{n+1}) + O(mn^2). \quad (\text{B12b})$$

⁸Precisely, the time per sample is $O(1)$ after a preprocessing time of $O(m)$. In our use case, we need a single sample from each pmf, so we characterize the time as just the total $O(m)$.

In (B12a), we use the fact that $k \leq n$ to bound $k2^k = O(n2^k)$. The summation $\sum_{k=1}^n 2^k = 2(2^n - 1)$, and together these give the term $O(n2^{n+1})$ in (B12b). Algorithm CC-C generates a sample in time $O(n2^n) + O(mn^2)$, requiring space $O(m)$. To our knowledge, CC-C is currently the best algorithm to simulate boson sampling in the regime with no collisions and no additional constraints, i.e., on the depth of the circuit.

APPENDIX C: ALGORITHM CP

In this section, we elaborate on the Cifuented and Parrilo [10] algorithm, denoted CP, from Sec. III B, particularly the details of the dynamic programming tables, and the running time. In the following, we write $\text{per}_M(R, C) = \text{per} M|_{R,C}$ for the permanent of a submatrix of M with rows R and columns C , following the notation of Ref. [10].

1. Dynamic programming tables

a. The local Q tables

For every node $t \in T$, we have the table of local permanents $Q[t](R, C) = \text{per}_M(R, C)$ for all $R \subseteq \rho(t)$ and $C \subseteq \kappa(t)$, i.e., resulting from combinations of row and column vertices contained in the node t . These represent submatrices of M with those rows and columns.

b. The subtree P tables

Crucial in the construction are the tables of subtree permanents $P[t](R, C)$ where $R \subseteq \rho(t)$ and $C \subseteq \kappa(t)$ again, same as in $Q[t]$. Their values are

$$P[t](R, C) = \text{per}_M(R \cup \bar{\Delta}_t^\rho, C \cup \bar{\Delta}_t^\kappa), \quad (\text{C1})$$

where $\bar{\Delta}_t^\rho := \rho(T_t) \setminus \rho(t)$, and similarly $\bar{\Delta}_t^\kappa := \kappa(T_t) \setminus \kappa(t)$. The table $P[t]$ stores the permanents of submatrices represented by the entire subtree T_t . Note that the labels of $P[t]$ contain only rows $R \subseteq \rho(t)$ and columns $C \subseteq \kappa(t)$, but the table elements relate to permanents of submatrices with rows and columns coming from the descendants of t , ensured by $\bar{\Delta}_t^\rho$ and $\bar{\Delta}_t^\kappa$. For leaves ℓ , we have $P[\ell] = Q[\ell]$, because they have no descendants. Conversely, the root $r := \text{root}(T)$ contains the final result:

$$P[r](\rho(r), \kappa(r)) = \text{per}_M(\rho(T), \kappa(T)) = \text{per} M. \quad (\text{C2})$$

In (C2), the equality follows from the Axiom (T1), which says that $\rho(T)$ is the set of all rows of the (whole) matrix M , and similarly $\kappa(T)$ is the set of all columns.

Note C.1. We remark on the notational distinction between $\bar{\Delta}_t^\rho$ ($\bar{\Delta}_t^\kappa$) in this section and $\Delta_{c_j}^\rho$ ($\Delta_{c_j}^\kappa$) in the pseudocode in Fig. 5 in Sec. III B. While the former (with bar) corresponds to the entire subtree of t , the latter (without bar) corresponds to individual children of t and is defined in the following section.

c. Helper tables

The table $P[t]$ of an internal node t is computed using the P tables of the children of t in a subset convolution. This requires computing the helper tables shown below; however, these do not need to be stored for future computation. The first table $Q'[t|c_j]$ is associated with permanents of submatrices made of rows and columns in t that are shared with its child

c_j :

$$Q'[t|c_j](R, C) = (-1)^{|R|} Q[t](R, C), \quad (\text{C3})$$

where $R \subseteq \Delta_{c_j}^\rho := \rho(c_j) \cap \rho(t)$ and $C \subseteq \Delta_{c_j}^\kappa := \kappa(c_j) \cap \kappa(t)$.

The table $Q''[t \leftarrow c_j]$ brings permanents of submatrices from the child's table $P[c_j]$ up the tree and makes them available to node t . The values transferred are those that correspond to submatrices with rows and columns contained in c_j but not in t , together with a selection of rows $R \subseteq \Delta_{c_j}^\rho$ and columns $C \subseteq \Delta_{c_j}^\kappa$ shared by the two nodes. These index the table:

$$Q''[t \leftarrow c_j](R, C) = P[c_j](R \cup \Delta_{c_j}^\rho, C \cup \Delta_{c_j}^\kappa), \quad (\text{C4})$$

where $\Delta_{c_j}^\rho := \rho(c_j) \setminus \rho(t)$ and $\Delta_{c_j}^\kappa := \kappa(c_j) \setminus \kappa(t)$.

We shortly revisit Note C1. Recall the P tables in (C1) reference rows $\bar{\Delta}_t^\rho$ (columns $\bar{\Delta}_t^\kappa$) corresponding to the whole subtree T_t . In (C4), the helper table $Q''[t \leftarrow c_j]$, the device used to connect t to its subtree, only references the sets $\Delta_{c_j}^\rho$ ($\Delta_{c_j}^\kappa$) corresponding to the immediate children of t . The reason is that the permanents stored in $P[t]$ are obtained recursively using the tree structure, computed from partial permanents stored in the tables $P[c_j]$ of the children c_j , which are computed before $Q''[t \leftarrow c_j]$. This justifies the definition (C4).

d. Subset convolution

The final step in computing a value of $P[t](R, C)$ for any internal node t (including the root) is to perform the subset convolution over R and C of all the prepared tables: the local table $Q[t]$, and the helpers $Q'[t|c_j]$ and $Q''[t \leftarrow c_j]$ of all children $c_j \in \text{ch}(t)$. The subset convolution, given in Eq. (C5) below, computes the value $P[t](R, C)$ as a summation over collections of subsets $\{R_i^{(i, \prime \prime)} \subseteq R\}_i$ and $\{C_i^{(i, \prime \prime)} \subseteq C\}_i$ that partition R and C , respectively. The formula is

$$\begin{aligned} P[t](R, C) = & \sum_{\{R_i^{(i, \prime \prime)}, C_i^{(i, \prime \prime)}\}_i} \left(Q[t](R_t, C_t) \right. \\ & \times \prod_{c_j \in \text{ch}(t)} Q'[t|c_j](R'_{c_j}, C'_{c_j}) Q''[t \leftarrow c_j] \\ & \left. \times (R''_{c_j}, C''_{c_j}) \right), \end{aligned} \quad (\text{C5})$$

which is labeled (9) in Sec. III B. Substituting the definitions of the helper tables, the above implements the subset convolution from [10, Lemma 10].

Below, we give a concrete characterization of the subsets $\{R_i^{(i, \prime \prime)}, C_i^{(i, \prime \prime)}\}_i$, but first we explain at high level the form of the equation. Each summand in (C5) involves all of the tables $Q[t]$, $Q'[t|c_j]$, and $Q''[t \leftarrow c_j]$ (for all children c_j of t), and the subsets $\{R_i^{(i, \prime \prime)}, C_i^{(i, \prime \prime)}\}_i$ index those tables. We combine values stored in the tables Q , Q' , Q'' for particular submatrices that are selected using the subsets $\{R_i^{(i, \prime \prime)}, C_i^{(i, \prime \prime)}\}_i$. These table values are used to compute terms of the permanents stored in $P[t](R, C)$.

Each partition of rows $\{R_i^{(i, \prime \prime)} \subseteq R\}_i$ consists of

$$R_t \sqcup R'_{c_1} \sqcup R''_{c_1} \sqcup \dots \sqcup R'_{c_k} \sqcup R''_{c_k} = R,$$

where k is the number of children of t . Similarly, each partition of columns $\{C_i^{(t'')}\} \subseteq C_i$ consists of

$$C_t \sqcup C'_{c_1} \sqcup C''_{c_1} \sqcup \dots \sqcup C'_{c_k} \sqcup C''_{c_k} = C.$$

The constraints on what those individual subsets may be are given by the following: Let S be one of the above tables (e.g. $Q'[t|c_j]$ for some child c_j). If for some $\hat{R} \subseteq R$ and $\hat{C} \subseteq C$, the value $S(\hat{R}, \hat{C})$ is not defined, we assume $S(\hat{R}, \hat{C}) = 0$, or, equivalently, we exclude the corresponding partitions from the summation. This reasoning gives us the following constraints on the subsets $\{R_i^{(t'')}, C_i^{(t'')}\}_i$ in the partitions for possibly nonzero terms of (C5): Recall that $Q[t]$ is defined for sets of rows and columns contained in t , so

$$R_t \subseteq \rho(t) \quad \text{and} \quad C_t \subseteq \kappa(t).$$

Similarly, $Q'[t|c_j]$ and $Q''[t \leftarrow c_j]$ are defined for

$$R'_{c_j}, \bar{R}''_{c_j} \subseteq \Lambda_{c_j}^\rho \quad \text{and} \quad C'_{c_j}, \bar{C}''_{c_j} \subseteq \Lambda_{c_j}^\kappa,$$

respectively.

2. Running time details of CP

We now derive the running time of algorithm CP, following Ref. [10, Lemma 2, Theorem 12]. In the following, we have a matrix $M \in \mathbb{C}^{n \times n}$, a tree decomposition $T = (T, \rho, \kappa)$ of its graph $G(M)$, and this tree decomposition has treewidth ω .

Lemma C.2. For a node $t \in T$, computing the table $Q[t]$ takes time $O(2^\omega \omega^2)$.

Proof. (See Lemma 2 in Ref. [10].) We compute the table $Q[t]$ for pairs of subsets $R \subseteq \rho(t)$ and $C \subseteq \kappa(t)$, with values $Q[t](R, C) = \text{per}_M(R, C)$ by a Laplace expansion:

$$\text{per}_M(R, C) = \sum_{c \in C} M_{r,c} \text{per}_M(R \setminus \{r\}, C \setminus \{c\}),$$

where $r \in R$ is a selected row (we choose $r = \min R$ for convenience). Note that $\text{per}_M(\emptyset, \emptyset) = 1$ and $\text{per}_M(R, C) = 0$ if the sizes do not match, i.e., $|R| \neq |C|$. Thus, we only need to compute permanents of square submatrices selected from the node t . We do this by starting with the 1×1 matrices, and then increase the size on each step using the Laplace expansion.

There are $O(2^{|\rho(t)|+|\kappa(t)|}) = O(2^\omega)$ values to compute: this is the size of the domain, although it is an upper bound because, as mentioned above, some of the entries are always zero. Noting that ω is the treewidth, any node $t \in T$ has the size of its contents $|\rho(t)| + |\kappa(t)| - 1 \leq \omega$. For each of those steps, we need $O(|\kappa(t)|) = O(\omega)$ operations—this is the number of columns in the summation. Finally, for each summand, we need $O(\omega)$ time to find the sets $R \setminus r, C \setminus c$ and look up the stored value of the corresponding permanent. Thus, the time is bounded as $O(2^\omega \omega^2)$. ■

Lemma C.3. For an internal node $t \in T$, computing the table $P[t]$ takes time $O(k_t 2^\omega \omega^2)$, where k_t is the number of children of t .

Proof. (See Theorem 12 in Ref. [10].) First note that the helper tables $Q'[t|c_j]$ and $Q''[t \leftarrow c_j]$ for each child c_j of t simply reference $Q[t]$ and $P[c_j]$, in the latter case with an inexpensive set union in the argument. The relevant computation happens in the subset convolution, which is done in time

$O(k_t 2^\omega \omega^2)$, where Cifuentes and Parrilo adapt the method from Ref. [24]. ■

Theorem C.4 (running time of CP). We can compute $\text{per}M$ in time $O(n 2^\omega \omega^2)$.

Proof. (See Theorem 15, and also Theorem 5 for more context, both in Ref. [10].) To compute the permanent from the whole tree, we have to compute the tables $Q[t]$ and then $P[t]$ for all $t \in T$, which takes the times obtained from Lemmas C2 and C3 for each node.

This gives $O(|T| 2^\omega \omega^2)$ for the Q tables. We bound $|T| = O(n)$, i.e., the size of the tree is linear in the number of columns (and rows) of the matrix. This is an assumption that we choose a reasonable tree decomposition: we could have a much larger tree, but at that point, it would contain a lot of redundancy. Note that in every tree decomposition used in this paper, it is indeed the case that $|T| = O(n)$.

For the P tables, we also note the following: $\sum_{t \in T} k_t = |T| - 1$, i.e., summing the numbers of all children gives the number of nonroot nodes: this is because the root is not a child. Then

$$\begin{aligned} \sum_{t \in T} O(k_t 2^\omega \omega^2) &= O\left(2^\omega \omega^2 \sum_{t \in T} k_t\right) = O(|T| 2^\omega \omega^2) \\ &= O(n 2^\omega \omega^2). \end{aligned}$$

Thus, we have a bound on the time taken to compute all Q and P tables, and hence the permanent. ■

APPENDIX D: NEAREST-NEIGHBOR INTERFEROMETERS AND BANDED MATRICES

Figure 6 depicts the interferometer $\mathcal{U}$ composed of D alternating layers of two-mode gates (left-hand side, LHS), where successive layers are shifted to allow the creation of a causal cone. We show the resulting graph $G(\mathcal{U})$ in the example case of $D = 2$ on the RHS. Observe that the nearest-neighbor interaction and the alternation of different layers impose that w_1, w_2 can only increase by at most one for each added layer. This is consistent with the result in Ref. [20] that depth $D = m$ is required to generate an arbitrary unitary $\mathcal{U}$.

Lemma D.1 (bandwidth). The bandwidth w of the unitary of a circuit composed of D alternating layers, as shown in Fig. 6, satisfies the condition:

$$w = w_1 + w_2 + 1 \leq 2D. \quad (\text{D1})$$

Note that (D1) is an inequality because some beam splitters may act as a diagonal matrix, producing no coupling between the modes. This happens when the coupling angle $\theta = 0$ or π .

Proof. We use the help of Fig. 6, and we show how the widths grow as we increase the depth D by adding the appropriate beam splitter layers. We proceed by induction on D . For simplicity, we assume generic coupling angles, i.e., ignoring the cases where $\theta = 0, \frac{\pi}{2}, \pi, \dots$, because we are interested in the maximum possible connectivity between input and output modes. Likewise, we ignore the edge cases of 1×1 identity blocks at the first or last mode. These simplifications allow us to focus on the main idea, but do not obscure the full picture.

a. Base case. $D = 1$. A single layer is, by construction, a matrix with 2×2 blocks on the diagonal. In Fig. 6, in the first (rightmost) layer of the LHS, observe that input modes

are partitioned into pairs of nearest neighbors, and coupling is only possible within these pairs. Thus, the output modes of the layer are also partitioned correspondingly. Conclude that the maximum number of nonzero elements in a row or column is $w = 2 = 2D$.

b. Induction hypothesis. Suppose for D layers, we have bandwidth $w \leq 2D$. We add a new layer that is of a different type than the previous. It also acts on disjoint pairs of nearest-neighbor modes, but it is shifted by one mode in comparison to the previous layer. Observe in Fig. 6 that this, in general, expands the maximum reach of any given input mode by two more output modes: one upward (lower index) and one downward (higher index),⁹ i.e., the new bandwidth is bounded as

$$w' \leq w + 2 \leq 2(D + 1).$$

Together, paragraphs *a* and *b* prove the Lemma by induction. ■

APPENDIX E: TREE DECOMPOSITION MANIPULATIONS

1. Permutations of columns

In Sec. IV C 1, we show how to implement a column permutation on the tree decomposition. We prove below that this is correct:

Lemma E.1 (column permutation). Let $T = (T, \rho, \kappa)$ be a tree decomposition corresponding to (the graph of) matrix M . Let α be a permutation, and let M' be a matrix of the same dimension as M with components $M'_{i,j} = M_{i,\alpha(j)}$. This has the same columns as M , but their order is given by α .

We construct a tree decomposition $T' = (T, \rho, \kappa')$ of M' with the same tree and row function by permuting the column labels for each $t \in T$:

$$\kappa'(t) := \{\langle \alpha^{-1}(j) | \mid \langle j | \in \kappa(t) \rangle\}.$$

Proof. We show this in terms of $G(M')$, the bipartite graph of the matrix M' , which comes from the view of M' as a sum of outer products; see Fig. 2. The permuted matrix is

$$\begin{aligned} M' &= \sum_{i,j} M'_{i,j} |i\rangle \langle j| = \sum_{i,j} M_{i,\alpha(j)} |i\rangle \langle j| \\ &= \sum_{i,j} M_{i,j} |i\rangle \langle \alpha^{-1}(j)|. \end{aligned}$$

Permuting the columns of M to obtain M' is equivalent to renaming the vertices of $G(M)$ corresponding to columns by α^{-1} while keeping the connectivity and edge weights the same. This action is a graph isomorphism. Renaming those same vertices as contents of the tree decomposition T to obtain T' ensures that T' is a tree decomposition of $G(M')$. ■

2. Restriction of a tree decomposition

As introduced in Sec. IV C 2, and Figs. 8(a) and 8(b), we implement the taking of submatrices as restrictions of tree

decompositions, intuitively, by deleting labels from within the nodes of the tree. In this section, we show the formal details:

Definition E.2 (tree restriction). Let $T = (T, \rho, \kappa)$ be a tree decomposition of the graph $G(\mathcal{U}) = (V = \mathcal{R} \sqcup \mathcal{C}, E)$, and let $\mathcal{R}' \subseteq \mathcal{R}$ and $\mathcal{C}' \subseteq \mathcal{C}$ be subsets of rows and columns, respectively. Define the restriction of T to those subsets as $T|_{\mathcal{R}', \mathcal{C}'} = (T, \rho', \kappa')$. It has the same tree T , but the contents of its nodes are given by $\rho' : T \rightarrow 2^{\mathcal{R}'}$ and $\kappa' : T \rightarrow 2^{\mathcal{C}'}$ which are defined as $\rho'(t) = \rho(t) \cap \mathcal{R}'$ and $\kappa'(t) = \kappa(t) \cap \mathcal{C}'$ for all $t \in T$.

This tree restriction is a valid representation of the equivalent restriction of the matrix, that is, the submatrix with the selected rows and columns. Formally, we have the following lemma:

Lemma E.3. In the above notation, the restriction $T|_{\mathcal{R}', \mathcal{C}'}$ is a tree decomposition of the submatrix $\mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$ of $\mathcal{U}$ that has rows $\mathcal{R}'$ and columns $\mathcal{C}'$.

Proof. We have a tree decomposition $T = (T, \rho, \kappa)$ of $G(\mathcal{U})$, the graph of matrix $\mathcal{U}$. Let $\mathcal{R}' \subseteq \mathcal{R}$ be a subset of rows, and $\mathcal{C}' \subseteq \mathcal{C}$ of columns, and denote $\mathcal{U}' = \mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$. We show that $T|_{\mathcal{R}', \mathcal{C}'}$ is a valid tree decomposition of $G(\mathcal{U}')$.

Recall that both T and $T|_{\mathcal{R}', \mathcal{C}'}$ have the same tree T , but the contents of nodes in $T|_{\mathcal{R}', \mathcal{C}'}$ are defined as $\rho'(t) = \rho(t) \cap \mathcal{R}'$ and $\kappa'(t) = \kappa(t) \cap \mathcal{C}'$ for all $t \in T$.

a. First, check (T1) requiring that all vertices are included in the tree decomposition:

$$\begin{aligned} \bigcup_{t \in T} \rho'(t) &= \bigcup_{t \in T} [\rho(t) \cap \mathcal{R}'] = \left[\bigcup_{t \in T} \rho(t) \right] \cap \mathcal{R}' \quad (\text{E1a}) \\ &\stackrel{(\text{T1})}{=} \mathcal{R} \cap \mathcal{R}' \stackrel{(\mathcal{R}' \subseteq \mathcal{R})}{=} \mathcal{R}'. \quad (\text{E1b}) \end{aligned}$$

We omit the analogous steps to show $\bigcup_t \kappa'(t) = \mathcal{C}'$. Thus, we conclude that $T|_{\mathcal{R}', \mathcal{C}'}$ satisfies Axiom (T1) with respect to $G(\mathcal{U}')$.

b. Now, we show that $T|_{\mathcal{R}', \mathcal{C}'}$ satisfies (T2) with respect to $G(\mathcal{U}')$, requiring that each edge of the latter be contained in some tree node. Note that $G(\mathcal{U}')$ is an induced subgraph of $G(\mathcal{U})$, meaning that, for each edge $(|i\rangle, \langle j|)$ in $G(\mathcal{U})$, if its endpoints are in $G(\mathcal{U}')$, then the edge is in $G(\mathcal{U}')$ as well (with the same weight). By (T2) on T with respect to $G(\mathcal{U})$, for every edge $(|i\rangle, \langle j|)$, there is a node $t \in T$ containing it, i.e., $|i\rangle \in \rho(t)$ and $\langle j| \in \kappa(t)$. If $|i\rangle \in \mathcal{R}'$ and $\langle j| \in \mathcal{C}'$, then by definition of the restriction, they belong to $\rho'(t)$ and $\kappa'(t)$. Hence (T2) holds for $T|_{\mathcal{R}', \mathcal{C}'}$ with respect to $G(\mathcal{U}')$.

c. We finish by showing that (T3) holds for $T|_{\mathcal{R}', \mathcal{C}'}$ with respect to $G(\mathcal{U}')$, requiring that a set of nodes containing a vertex forms a subtree. Define the preimage of ρ as $\rho^{-1} : \mathcal{R} \rightarrow 2^T$ where $\rho^{-1}(|i\rangle) = \{t \in T \mid |i\rangle \in \rho(t)\}$ for all $|i\rangle \in \mathcal{R}$. Define the preimage κ^{-1} analogously. These give us the set of nodes that contain a row or column. By Axiom (T3) that holds for T with respect to $G(\mathcal{U})$, each $\rho^{-1}(|i\rangle)$ [respectively, $\kappa^{-1}(\langle j|)$] forms a subtree of T . Under the restriction to $\mathcal{R}'$ and $\mathcal{C}'$, the preimages are clearly just restrictions (on the domain) $(\rho')^{-1} = \rho^{-1}|_{\mathcal{R}'}$ and $(\kappa')^{-1} = \kappa^{-1}|_{\mathcal{C}'}$. Thus, we conclude that Axiom (T3) is satisfied for $T|_{\mathcal{R}', \mathcal{C}'}$ as a tree decomposition of $G(\mathcal{U}')$.

We conclude that the restriction $T|_{\mathcal{R}', \mathcal{C}'}$ is a valid tree decomposition, and that it is the decomposition of $G(\mathcal{U}')$, the graph of the submatrix $\mathcal{U}' = \mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$. ■

⁹We are ignoring the edge case when there are no more modes to expand into. Of course, the inequality in bandwidth still holds.

An important fact, stated in Eq. (15), is that the restriction produces a tree decomposition that is no worse than the original:

Lemma E.4 (treewidth). Using previous notation, the treewidth of the restriction is bounded as

$$\text{tw}(T|_{\mathcal{R}', \mathcal{C}'}) \leq \min(\text{tw}(T), |\mathcal{R}'| + |\mathcal{C}'| - 1).$$

Proof. A restriction may only remove vertices from the decomposition, so the treewidth can only decrease, i.e., $\text{tw}(T|_{\mathcal{R}', \mathcal{C}'}) \leq \text{tw}(T)$. Treewidth measures the size of the largest node, and that cannot be larger than $|\mathcal{R}'| + |\mathcal{C}'|$, i.e., the total number of vertices left. Recall from (8a) that the treewidth is one less than the size of the largest node. In general, we cannot know which of the choices $\text{tw}(T)$ or $|\mathcal{R}'| + |\mathcal{C}'| - 1$ will be lower, hence we keep the minimum operator. ■

Finally, we note that the tree restriction $T|_{\mathcal{R}', \mathcal{C}'}$ is not necessarily the best decomposition of $\mathcal{U}|_{\mathcal{R}', \mathcal{C}'}$: Lemma E.4 only shows the upper bound on its treewidth. However, crucially, the restriction is easy to find even when an optimal decomposition may not be, i.e., due to column permutations, or if we need a submatrix of a submatrix of $\mathcal{U}$. Furthermore, as the restriction requires nothing more than taking two set intersections at each node of the tree, it is efficient to compute.

3. Redundancy of nodes

As defined in Sec. IV B, each node of our tree decomposition contains a single column label. When restricting the decomposition to $T' = (T, \rho', \kappa')$, we may end up with a node $t \in T$ without any columns, i.e., $\kappa'(t) = \emptyset$. Indeed, our algorithm will perform such restrictions extensively. In the following Lemma E.6, we prove that in this case, deleting this node and connecting its neighbors in the appropriate way leads to a new tree decomposition, denoted $T \setminus t$, that still gives the correct permanent.

Definition E.5 (removing a node). Let T be a tree decomposition with a linear tree T (no branching), as is the case in our algorithm, let t be an internal node (not root nor leaf), and let p, c be the parent and the single child of t , respectively. Then $T \setminus t$ is a decomposition without the node t where p is connected directly to c . If t is the root (has no parent), then delete it and make its child c the new root in $T \setminus t$. If t is a leaf, we just delete it. We present an example in Figs. 8(c) and 8(d).

Below, we show that this preserves the permanent by proving that both T and $T \setminus t$ represent the same matrix if $\kappa'(t) = \emptyset$. Intuitively, in terms of Algorithm CP, the tables $Q[t]$ and $Q'[t|c_j]$ of node t (and its child c_j) become trivial, and the tables $Q''[t \leftarrow c_j]$ and $P[t]$ only act to bring data from the child of t up the tree, making them accessible to the parent of t . Thus, we may use $T \setminus t$ instead of T in our permanent calculation.

Lemma E.6 (redundant nodes). Let $T = (T, \rho, \kappa)$ be a tree decomposition of the graph of a matrix M , where T is a linear

tree. Suppose there is a node $t \in T$ such that $\kappa(t) = \emptyset$ and $\rho(t) \subseteq \rho(T \setminus t)$. Then $T \setminus t$ also represents the matrix M .

Note E.7. A subtle but important formal detail is the condition that $\rho(t) \subseteq \rho(T \setminus t)$. The decomposition $T \setminus t$ is not equivalent to T for computing the permanent if there exists a row vertex $|i\rangle \in \rho(t)$ that is not contained in any other node. If such a row vertex $|i\rangle$ exists, and since $\kappa(t) = \emptyset$, there is no column vertex to which $|i\rangle$ could be connected. This corresponds to having a row of all zeros in the matrix and consequently $\text{per} M = 0$. In such a case, using $T \setminus t$ we would compute the wrong (generally nonzero) permanent.

The above problem can arise if we obtain T as a restriction of some larger tree decomposition, in which case we have to be careful: Suppose there is a column $\langle j|$ that has a single nonzero value in row $|i\rangle$. A restriction that removes $\langle j|$ must also remove $|i\rangle$.

Proof of Lemma E.6. We verify that $T \setminus t$ satisfies the axioms of a tree decomposition with respect to the graph $G(M)$. Recall that $G(M)$ has vertices $\mathcal{R} \sqcup \mathcal{C}$, where $\mathcal{R}$ is the set of row labels and $\mathcal{C}$ of column labels. There is an edge $(|i\rangle, \langle j|)$ if and only if M_{ij} is nonzero.

a. In the original tree decomposition T , by Axiom (T1), $\rho(T) = \bigcup_{q \in T} \rho(q) = \mathcal{R}$ and similarly $\kappa(T) = \mathcal{C}$. In our case, node t has no columns [$\kappa(t) = \emptyset$], so $\kappa(T \setminus t) = \mathcal{C}$ already. Removing t does not violate Axiom (T1) on columns.

We verify the same for rows: We have $\rho(T) = \mathcal{R}$ and $\rho(t) \subseteq \rho(T \setminus t)$. This implies that

$$\mathcal{R} = \rho(T) = \rho(t) \cup \rho(T \setminus t) = \rho(T \setminus t).$$

Thus $T \setminus t$ satisfies Axiom (T1).

b. Now, verify Axiom (T2), stating that every edge must be contained in a node. The decomposition T satisfies this. Node t has $\kappa(t) = \emptyset$, so it cannot contain any edges. Therefore, all edges are contained in $T \setminus t$, and thus the latter satisfies Axiom (T2).

c. Finally, to verify Axiom (T3) requiring that the set of all nodes containing a row (respectively, column) forms a subtree of T , we note that this is already the case in T , and that removing a node cannot invalidate this. ■

Example E.8 (disjoint graphs). As an illustrative special case, suppose that t has no columns as above, and also no rows, i.e., $\rho(t) = \emptyset$. This clearly satisfies the conditions of Lemma E.6. Denote T_1 the set of ancestors of t and T_2 the set of descendants of t , both subtrees of T . Since t is completely empty, by Axiom (T3), the two trees T_1 and T_2 correspond to blocks of the matrix which share no rows nor columns. We can also view this as joining together two independent graphs (and their tree decompositions) by putting an empty node between them. Up to an appropriate permutation of rows and columns, corresponding to a change of basis, the matrix is block diagonal, and its permanent is the product of permanents of blocks.

[1] S. Aaronson and A. Arkhipov, The computational complexity of linear optics, *Theory Comput.* **9**, 143 (2013).

[2] J. Preskill, Quantum computing and the entanglement frontier, [arXiv:1203.5813](https://arxiv.org/abs/1203.5813).

- [3] L. Valiant, The complexity of computing the permanent, *Theor. Comput. Sci.* **8**, 189 (1979).
- [4] H. J. Ryser, *Combinatorial Mathematics*, *Carus Mathematical Monographs* (Mathematical Association of America; distributed by John Wiley and Sons, Inc., New York, 1963), Vol. 14, pp. xiv+154.
- [5] D. G. Glynn, The permanent of a square matrix, *Eur. J. Comb.* **31**, 1887 (2010).
- [6] P. Clifford and R. Clifford, The classical complexity of boson sampling, in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18 (Society for Industrial and Applied Mathematics, USA, 2018), pp. 146–155.
- [7] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu, P. Hu, X.-Y. Yang, W.-J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, Z. Wang, *et al.*, Quantum computational advantage using photons, *Science* **370**, 1460 (2020).
- [8] H.-S. Zhong, Y.-H. Deng, J. Qin, H. Wang, M.-C. Chen, L.-C. Peng, Y.-H. Luo, D. Wu, S.-Q. Gong, H. Su, Y. Hu, P. Hu, X.-Y. Yang, W.-J. Zhang, H. Li, Y. Li, X. Jiang, L. Gan, G. Yang, L. You, *et al.*, Phase-programmable Gaussian boson sampling using stimulated squeezed light, *Phys. Rev. Lett.* **127**, 180502 (2021).
- [9] R. García-Patrón, J. J. Renema, and V. Shchesnovich, Simulating boson sampling in lossy architectures, *Quantum* **3**, 169 (2019).
- [10] D. Cifuentes and P. A. Parrilo, An efficient tree decomposition method for permanents and mixed discriminants, *Lin. Alg. Appl.* **493**, 45 (2016).
- [11] C. Oh, Y. Lim, B. Fefferman, and L. Jiang, Classical simulation of boson sampling based on graph structure, *Phys. Rev. Lett.* **128**, 190501 (2022).
- [12] H. Qi, D. Cifuentes, K. Brádler, R. Israel, T. Kalajdziewski, and N. Quesada, Efficient sampling from shallow Gaussian quantum-optical circuits with local interactions, *Phys. Rev. A* **105**, 052412 (2022).
- [13] A. E. Moylett and P. S. Turner, Quantum simulation of partially distinguishable boson sampling, *Phys. Rev. A* **97**, 062329 (2018).
- [14] P. Kok and B. W. Lovett, *Introduction to Optical Quantum Information Processing* (Cambridge University Press, Cambridge, UK, 2010).
- [15] G. Voigt, Tree decompositions, treewidth, and NP-hard problems, Mathematics, Massachusetts Institute of Technology (2016), term paper in the course 18.204 Undergraduate Seminar in Discrete Mathematics.
- [16] J. Carolan, C. Harrold, C. Sparrow, E. Martín-López, N. J. Russell, J. W. Silverstone, P. J. Shadbolt, N. Matsuda, M. Oguma, M. Itoh, G. D. Marshall, M. G. Thompson, J. C. F. Matthews, T. Hashimoto, J. L. O'Brien, and A. Laing, Universal linear optics, *Science* **349**, 711 (2015).
- [17] B. A. Bell and I. A. Walmsley, Further compactifying linear optical unitaries, *APL Photonics* **6**, 070804 (2021).
- [18] R. A. Campos, B. E. A. Saleh, and M. C. Teich, Quantum-mechanical lossless beam splitter: SU(2) symmetry and photon statistics, *Phys. Rev. A* **40**, 1371 (1989).
- [19] R. Jozsa, On the simulation of quantum circuits, *arXiv:quant-ph/0603163*.
- [20] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, Optimal design for universal multiport interferometers, *Optica* **3**, 1460 (2016).
- [21] B. Go, C. Oh, L. Jiang, and H. Jeong, Exploring shallow-depth boson sampling: Towards scalable quantum supremacy, *Phys. Rev. A* **109**, 052613 (2024).
- [22] S. Novák, Efficient simulation of shallow quantum circuits: Boson sampling, UG4 Project and PGT Dissertation Public Archive, School of Informatics, University of Edinburgh (2022).
- [23] A. J. Walker, New fast method for generating discrete random numbers with arbitrary frequency distributions, *Electron. Lett.* **10**, 127 (1974).
- [24] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, Fourier meets Möbius: Fast subset convolution, in *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07 (Association for Computing Machinery, New York, 2007), pp. 67–74.