

Boundaries for quantum advantage with single photons and loop-based time-bin interferometers

Samo Novák¹, David D. Roberts¹, Alexander Makarovskiy¹, Raúl García-Patrón^{2,3}, and William R. Clements¹

¹ORCA Computing, London, UK

²Laboratory for the Foundations of Computer Science, School of Informatics, University of Edinburgh

³Phasecraft Ltd., London, UK

25 November 2024

Loop-based boson samplers interfere photons in the time degree of freedom using a sequence of delay lines. Since they require few hardware components while also allowing for long-range entanglement, they are strong candidates for demonstrating quantum advantage beyond the reach of classical emulation. We propose a method to exploit this loop-based structure to more efficiently classically sample from such systems. Our algorithm exploits a causal-cone argument to decompose the circuit into smaller effective components that can each be simulated sequentially by calling a state vector simulator as a subroutine. To quantify the complexity of our approach, we develop a new lattice path formalism that allows us to efficiently characterize the state space that must be tracked during the simulation. In addition, we develop a heuristic method that allows us to predict the expected average and worst-case memory requirements of running these simulations. We use these methods to compare the simulation complexity of different families of loop-based interferometers, allowing us to quantify the potential for quantum advantage of single-photon Boson Sampling in loop-based architectures.

1 Introduction

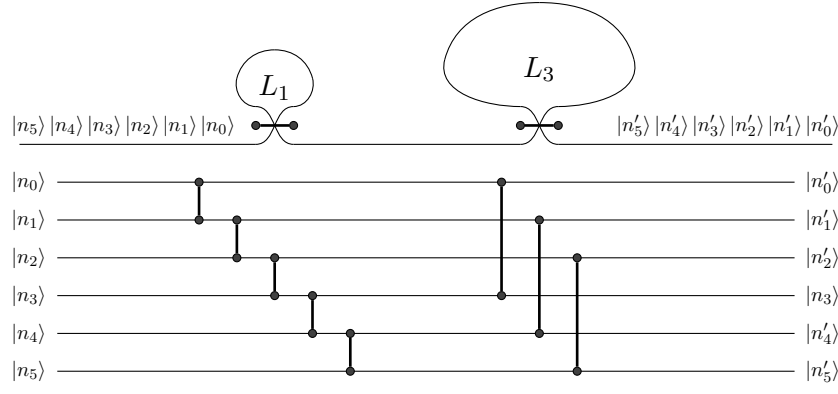
Boson sampling is a promising candidate in the search for quantum advantage because it presents a problem solvable by a quantum computer that

generally cannot be solved classically. In single-photon Boson Sampling (SPBS) proposed by Aaronson & Arkhipov [1], a multi-mode Fock state made of a collection of single photons is sent through a passive interferometer and measured by photon number detectors. The probability distribution governing this process depends on the permanent of a matrix, the computation of which is $\#P$ -hard [2]. Under reasonable assumptions, sampling from such a distribution is classically hard [1]. As such, boson sampling is generally intractable to simulate using a classical computer beyond a few tens of photons. However, a quantum device can generate samples efficiently.

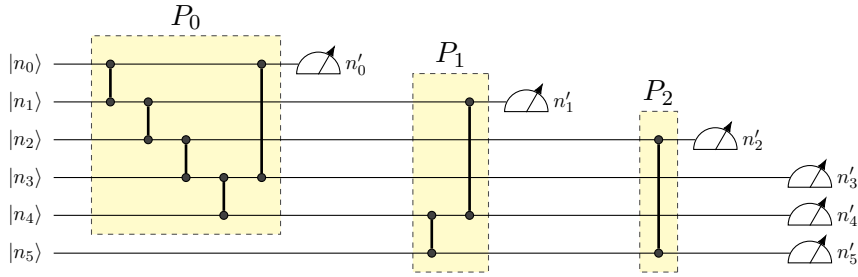
To show that a specific SPBS system achieves quantum advantage, we must determine the complexity of classically simulating the sampling process using currently known methods, and show that this is beyond the reach of current classical computers. In order to achieve sufficiently high simulation complexity, we need the interferometer to allow a large number of modes to interfere with each other, generally requiring many components to build, for example the triangular [3] or rectangular [4] universal schemes. If the connectivity is too low, the entanglement is local and the system can be efficiently simulated, e.g. by tensor network methods [5].

However, one family of architectures that allows such connectivity but are easier to build are loop-based systems, used for example by Madsen *et al.* [6] in an experiment that claimed to achieve quantum advantage. These architectures work in the time degree of freedom where modes, called time-bins, correspond to regular time intervals in which photons arrive [7, 8]. The inter-

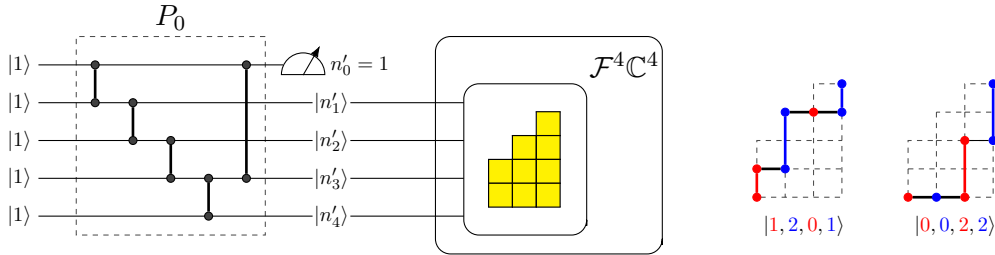
William R. Clements: wclements@orcacomputing.com



(a) A system made of loops $L_1 \circ L_3$



(b) Progressive decomposition of $L_1 \circ L_3$



(c) State space of possible outputs after the zeroth component P_0 of $L_1 \circ L_3$ and measurement $n'_0 = 1$

Figure 1: Overview of the results. (a) A time-bin boson sampler consisting of loops of different lengths composed in sequence. The input photons arrive in regular intervals, represented by time-bin modes with photon numbers $|n_0, n_1, \dots, n_{m-1}\rangle$. The loops couple earlier time-bin modes to later ones, and the lengths of the loops are given in time-bin units: thus a loop of length ℓ allows photons in modes a and $a + \ell$ to interfere. This is shown below the physical schematic as a circuit. Note that by convention, the first ℓ time-bins are deterministically loaded into a loop of length ℓ , and we omit drawing the initial vacuum modes inside the loop. (b) *Progressive decomposition* of the circuit, where each yellow box P_a contains only beamsplitters in the causal cone of the measurement of mode a , except for those already captured by previous causal cones. This decomposition allows us to evolve the wavefunction only through components P_a that all have a bounded effective number of modes, measure their top mode, and collapse the wavefunction into a subspace. (c) An example of our lattice path formalism that describes the reachable state space of a loop-based system throughout simulation. Here, we show the state space after evolving the input wavefunction $|1^5\rangle$ through component P_0 and measuring 1 photon in the top mode, thus collapsing the wavefunction into a subspace shown as the yellow lattice diagram. Each basis state $|n'_1, n'_2, n'_3, n'_4\rangle$ of the space that can be found with nonzero probability corresponds to a path within the yellow diagram that starts in the bottom left corner, ends at top right corner, and takes integer steps up or right. The numbers of photons correspond to vertical steps taken by a path, as shown for two example paths on the right-hand-side. Basis states of the full Fock space $\mathcal{F}^4\mathbb{C}^4$ not contained in the yellow diagram are not reachable.

ferometer consists of time-delay loops of various lengths, such as the ones illustrated in Figure 1a, which allow photons in distinct time-bin modes to interfere.

Investigation of the complexity of simulating such loop-based systems is needed because, while general SPBS is known to be hard, the restricted circuit topology may be exploited to speed up classical simulation. This would raise the bar for quantum advantage in such loop-based systems. Recent examples include Lubasch *et al.* [8], who focused on the case where all loops have the same length, and Deshpande *et al.* [9] who showed that two loops of different lengths are also not complex enough for advantage.

However, in the case of more general loop-based systems, further study is required to determine if and when these systems are simulable or not. Our work advances this field of study by establishing new conditions on the sequence of delay lines that should be used to achieve advantage. Our main contributions are described in the following.

Using a representation of a loop as a staircase-shaped sequence of possibly nonlocal gates, as illustrated in Figure 1a,¹ we highlight the limited connectivity of the interferometer. Following a light-cone argument, we decompose the circuit into smaller components shown in Figure 1b. Each component has a bounded effective number of modes, and only contains a small number of beamsplitters; thus *strongly* simulating a single component (by computing its state vector evolution) is easier. Our method measures a single mode after simulating each component, and collapses the wavefunction correspondingly, keeping the memory requirement tractable even for large systems. The wavefunction evolution through each component can be computed using any wavefunction simulator capable of working in the Fock (number) basis, giving a way for further specific optimizations or extensions.

To quantify the complexity of the above simulation method, and hence help in the search for quantum advantage, we develop a new formalism based on lattice paths that fully describes the state space throughout the simulation. An extension of the work of Brádler & Wallner [10], our

formalism maps one possible output state as one path in the lattice, and the entire reachable space corresponds to a region bounded by the maximal path; see an example in Figure 1c. Our formalism provides tools to reason not only about purely loop-based systems: one can use it to characterize the state space in an arbitrary circuit of beam-splitters and measurements, as long as the first operation is a loop of length one time-bin, and the input consists of Fock states.

The structure of our paper is the following: First, in Section 1.1, we compare our approach to existing methods found in the literature. In Section 2, we present the background material necessary to understand the paper. Then we study the circuit topology of loop-based systems, and use it to design a new simulation algorithm in Section 3. In Section 4 we develop our new lattice path-based formalism to characterize the state space, and in Section 5, we show how our formalism can be used to obtain the memory complexity statistics. We use these tools to evaluate the potential for quantum advantage in a family of loop-based systems where the lengths are $1, \ell, \ell^2$. Finally, in Section 6, we contextualize our findings and identify advantage candidates that warrant further study.

We provide a companion Git repository² with a reference implementation of the simulation algorithm, as well as the lattice-path computation and complexity estimation used to obtain results in Section 5.

1.1 Comparison with other approaches

First, we summarize how our approach compares to other relevant work in the literature. Our approach provides an efficient method for simulating loop-based circuits with a very large number of photons and modes, as long as there are few loops and their lengths are kept short. This regime is not covered by any other classical simulation algorithm to date. We describe these other algorithms in the following. Note that we focus here on exact sampling algorithms as the most relevant comparison, and do not consider methods that exploit noise to approximate boson sampling such as [15].

²<https://github.com/orcacomputing/loop-progressive-simulator>

¹Note we use the same convention as [7] where we preload the loops with the first time bins which leads to the circuit as shown, omitting the initial vacuum modes of the loops.

Circuit type $\Rightarrow$ Algorithm $\Downarrow$	power-law (Λ loops, base ℓ)		full-depth Haar-random triangular [3] or square [4]
our method (average instance)	$\ell = 1$	$\mathcal{O}(m\Lambda^3 2.6^\Lambda)$	$\mathcal{O}(m^4 2.6^m)$
	$\ell > 1$	$\mathcal{O}(m\Lambda \ell^{2(\Lambda-1)} 2.6^{\ell^{\Lambda-1}})$	n/a
Clifford & Clifford [11] incl. tree decompositions [12, 13]	$\mathcal{O}(m2^{m/2}) + \mathcal{O}(m^3)$		
SLOS_full [14]	$\mathcal{O}(m2.6^m)$		

Table 1: Computational scaling of our method compared to other simulation and sampling algorithms. Here m is the number of modes, and we consider systems where the number of photons is half the number of modes. For the power-law architecture, we consider a sequence of Λ loops of lengths $1, \ell, \ell^2, \dots, \ell^{\Lambda-1}$. The base of exponentials $2.6 \approx \sqrt{27}/2$ comes from the use of the Stirling’s formula to simplify $\binom{\frac{3}{2}x-1}{x}$ as x grows, see (C.5).

As a first point of reference, Clifford and Clifford [11] developed an algorithm generating a sample from an m -mode n -photon experiment in time $\mathcal{O}(n2^n) + \mathcal{O}(mn^2)$. This general method does not exploit any internal structure of the interferometer and only depends on the number of modes and photons. As such, it becomes impractical in regimes above 40 photons, even when the interferometer is highly structured. Our algorithm can sample from states with many more photons in simple loop-based systems.

For settings where local-only interactions induce a sparse graph structure, tree decomposition techniques can reduce the complexity of the Clifford and Clifford algorithm to $\mathcal{O}(mn^2\omega^2 2^\omega)$ [12], with a further reduction to $\mathcal{O}(n^2 2^\omega \omega^2) + \mathcal{O}(\omega n^3)$ achievable in the 1D lattice case [13]. Here ω corresponds to the (highest) treewidth of the graphs representing measurement outcomes. However, in general, these tree decomposition techniques are not transferrable to our setting: in the presence of loops of arbitrary lengths, long-range interference occurs which leads to less sparsity in the graph representing the connectivity of modes. As such, they cannot be used for the loop-based circuits considered in this work.

Another point of reference is the SLOS_full [14] *strong* state vector simulator. The SLOS_full algorithm achieves a running time $\mathcal{O}(n \binom{n+m-1}{m-1})$ where the binomial is the dimension of the Fock space. It is general in the sense that it does not exploit any internal structure. This, however, means that at some number of photons and modes, the time and memory requirements become intractable. Our algorithm, on the other hand, calls a strong simulation subroutine only on a smaller subcircuit, with a smaller number of photons, and then

samples some of the photons, thus keeping the problem tractable even for many photons and modes in a loop-based system. Note that on a Haar-random circuit, our method is slower than SLOS_full since we evolve the state vector over all the beam splitters one by one instead of just using the matrix description of the overall unitary to calculate the state.

In Table 1, we summarize the comparison in a table showing the time complexities of our algorithm and the other algorithms described above. As illustrative examples we focus on two cases. First, we consider the *power-law* architecture with the sequence of loops of lengths $1, \ell, \ell^2, \dots, \ell^{\Lambda-1}$ (with integer ℓ and Λ) that we study in Section 5.3, with an input state $|n\rangle = |1, 0, 1, 0, \dots\rangle$. Ours is the only method that does not scale exponentially with the number of photons or modes, with scaling instead with ℓ and Λ . We also contrast this with a Haar-random circuit, modelled as a triangular or rectangular schemes [3, 4] which can also be implemented in a loop-based system (setting $\ell = 1$), and the same input state. For these Haar-random circuits, other methods are more appropriate. The derivation of the complexities presented in this table can be found in Appendix C.

We note that the theoretical hardness of loop-based circuits has been explored by Go *et al.* [16, 17]. In [16], the authors present a circuit with a “non-local hypercubic structure” which can be implemented with a power-law architecture with sequential loops of length $1, 2, 4, \dots, 2^{\lfloor \log_2(m) \rfloor}$ where m is the number of modes. This circuit is shown to exhibit properties that are conducive to hardness, which include the ability to mimic a Haar-random unitary up to the first moment and the so-called *hiding* property exploited in

the usual hardness proofs for boson sampling. In [17], a circuit with two repetitions in sequence of this power-law architecture was shown to be hard. Our work bridges some of the gap between known classical simulation methods for one loop [8] and two loops [9] and the hardness results from Go *et al.* by extending the range of simulable loop-based architectures.

2 Background

We start by introducing some background to further motivate the study of loop-based time-bin interferometers and to introduce our new methods to simulate and characterize them. We give a short overview of SPBS in Section 2.1, we motivate and define loop-based time-bin interferometers in Section 2.2, and we present the background for the lattice path formalism in Section 2.3.

2.1 Single-photon Boson Sampling

Single-photon Boson Sampling (SPBS) is a quantum optical experiment defined by Aaronson & Arkhipov in [1] where we send a collection of n indistinguishable single photons into a linear interferometer on m modes (ports) and measure where they exit. The quantum state of light in such an experiment is described by a vector in Fock space $\mathcal{F}^n \mathbb{C}^m$. This is spanned by the number basis made of vectors $|n_0, \dots, n_{m-1}\rangle$ where each n_a is the number of photons in mode a . The dimension of a Fock space of n indistinguishable photons distributed in m modes is:

$$\dim \mathcal{F}^n \mathbb{C}^m = \binom{m-1+n}{n} \quad (2.1)$$

The SPBS problem is defined as follows: generate Fock basis states measured after sending an input $|\underline{n}\rangle$, usually with $n_a \leq 1$ for each mode, into a linear interferometer. The interferometer is a mode-to-mode transformation described by a unitary matrix $U : \mathbb{C}^m \rightarrow \mathbb{C}^m$. There are many ways to construct these, most commonly decomposing them as a system of single-mode phase-shifters and two-mode beamsplitters [3, 4]. We represent a beamsplitter $B(\theta) : \mathbb{C}^2 \rightarrow \mathbb{C}^2$ as the following rotation matrix [18]:

$$B(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}. \quad (2.2)$$

In a system of more than two modes, we specify that a beamsplitter acts on modes a, b as $B_{a,b}(\theta)$. Note that we omit any reflection or transmission phases, only representing the coupling angle θ . Furthermore, we will often care only about a coupling being possible but not the concrete value of θ , in which case we omit it from the notation.

The input-output transition amplitudes of the interferometer U acting on the Fock space are permanents of certain matrices derived from U [1]. The computation of a permanent of a general matrix belongs to the complexity class #P-hard [2, 1]. This means that classically simulating Boson Sampling is likewise hard, and that is the reason why Boson Sampling devices, which can generate these samples natively, are candidates for quantum advantage.

2.2 Loop-based time-bin interferometers

One specific type of architecture used for Boson Sampling, proposed by Motes *et al.* [7] and experimentally implemented for example by Madsen *et al.* [6] and Carosini *et al.* [19], is based on loops. Here, some output port of a device is connected back to an input port of the same device using an optical loop; see an example in Figure 1a. The modes of a loop-based interferometer, called *time-bins*, correspond to regular time intervals when photons arrive or leave [8]. This is useful because it allows us to build complex interferometers without using many physical components [20], and the combination of being practically realizable, but at the same time allowing large interferometers, makes loop-based architectures candidates for quantum advantage demonstrations.

2.2.1 Single loop

We define an optical loop as a device consisting of a delay line and a single beamsplitter which may change its transmissivity in time. The loop connects one output port of the beamsplitter to an input; the remaining two ports are the input and output of the device.

We understand this as a circuit consisting of a staircase-shaped sequence of beamsplitters, as in Figure 1a. The length of the loop is given in time-bin units, so if the loop has length ℓ , denoted L_ℓ , then it couples mode a to $a + \ell$. Note that for every loop L_ℓ , including when these are com-

posed in Section 2.2.2, we employ the convention of Motes *et al.* [7] where photons from the initial ℓ modes deterministically enter the loop, and we omit the latter's original vacuum modes. This gives us the circuit in Fig. 1a, and the following definition:

Definition 2.1 (optical loop). Let $\ell \geq 1$. Then define a loop of length ℓ as the following composite of time-bin beamsplitters:

$$L_\ell = B_{0,\ell} \circ B_{1,1+\ell} \circ \cdots \circ B_{a,a+\ell} \circ \cdots,$$

where the composition is written from left to right ($A \circ B := BA$ for maps A, B), and we omit the coupling angles. In practice, the total number of modes m is finite and the above composite stops at $B_{m-1-\ell, m-1}$.

The simplest example is the loop $L_1 = B_{0,1} \circ B_{1,2} \circ \cdots$, which looks like a staircase of nearest-neighbour beamsplitters when drawn as a circuit (see the left part of Figure 1a). This loop is important throughout the paper, because it establishes a shared pool of photons to be distributed between modes; we come back to this idea in Sections 2.3 and 4.

2.2.2 Sequential composites of loops

A single photonic loop L_ℓ is a fairly simple system, and in particular it can be efficiently simulated [8]. This means that in order to achieve quantum advantage, we need to look elsewhere, and the natural idea for a loop-based setup is to compose multiple loops together.

There are different ways that multiple loops could be joined together, and in our work we focus on *sequential* composites of loops $L_{\ell_1} \circ L_{\ell_2} \circ \cdots \circ L_{\ell_\Lambda}$, where Λ is the number of loops, and the lengths ℓ_j may be different. An example of such architecture is $L_1 \circ L_3$ shown in Figure 1a. Note that if all lengths $\ell_j = 1$, then the individual loops are similar to slices of the Reck scheme [3], a well-known universal decompositions of optical unitaries to a system of beamsplitters and phase shifters.

Definition 2.2 (power-law composites). An interesting family of sequential composites of loops are those with lengths $\underline{\ell} = (1, \ell, \ell^2, \dots, \ell^{\Lambda-1})$, which we call *power-law* composites. We call ℓ the *base loop length*.

Power-law systems are important because of their simple form that still allows long-range entanglement in shallow circuits (Λ small). The

property that each length is an exponent of ℓ generates connectivity between modes that can be understood as the volume of a Λ -dimensional hypercube with sides of length ℓ , where modes are vertices at integer coordinates and edges correspond to beamsplitters [9, 16]. Such architectures have been recently used in an advantage demonstration [6], and in Boson Sampling hardness results [17, 16].

2.3 Lattice path formalism for bosonic Fock spaces

The circuit topology of loop-based systems restricts where photons can go, and this can be captured by a formalism where the state space corresponds to a region in an integer lattice. Possible output number states correspond to lattice paths within this region (see for example Figure 1c), and thus the lattice-path representation can be used to count the state space using combinatorial methods.

In this section, we describe the original lattice-path formalism developed by Brádler & Wallner [10] that can be used to count and enumerate the state space of a system of loops that all have length one. We expose their results using a different notation and terminology that will suit our needs later; and we then expand their formalism to general loop-based systems in Section 4.

2.3.1 Lattice paths

Informally, a *lattice* is a collection of discrete points in space that repeat in regular intervals. We will be interested in lattices contained within finite rectangles in the $\mathbb{N}^2$ quadrant of $\mathbb{Z}^2$, and in the paths allowed within these rectangles.

Definition 2.3 (lattice path). Let $W, H \in \mathbb{N}$. A *lattice path* in the rectangle of dimensions (W, H) is a path in $\mathbb{N}^2$ that starts at the origin $(0, 0)$, ends at point (W, H) , and only takes steps right $(+1, 0)$ or up $(0, +1)$. The set of all (W, H) -lattice paths is denoted $\mathcal{L}(W, H)$.

To describe individual paths in $\mathcal{L}(W, H)$, we define two equivalent representations, each with a different domain of usefulness. We show an example path and both representations in Figure 2.

Definition 2.4 (height representation). In *height representation*, a path is written as a nondecreasing sequence of integers $\lambda = (\lambda_0, \dots, \lambda_W)$. Each

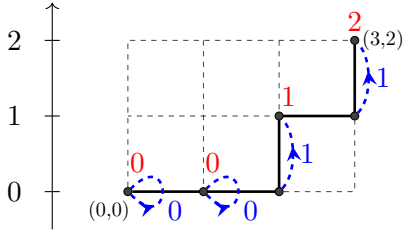


Figure 2: Example of a lattice path from $\mathcal{L}(3, 2)$ in both the height and step representations. **(red)** Height representation $\lambda = (0, 0, 1, 2)$ where the coordinates indicate the height of the path above the horizontal axis. This is the maximum y coordinate of a point passed by the path at each x coordinate. **(blue)** Step representation $\underline{n}^\lambda = \Delta\lambda = (0, 0, 1, 1)$ with coordinates $\underline{n}_a^\lambda = \lambda_a - \lambda_{a-1}$ that represent the size of the vertical steps the path takes, shown by blue dashed arrows.

component $0 \leq \lambda_a \leq H$ is defined as

$$\lambda_a = \max\{y_i : (a, y_i) \in \mathbb{N}^2 \text{ is passed by } \lambda\},$$

where (a, y) are points with $x = a$ passed by the path; see example in Figure 2 in red. Note that since a path must end at point (W, H) , the final component is always $\lambda_W = H$. Height representation is denoted by lowercase Greek letters $\lambda, \kappa, \mu, \dots$

Definition 2.5 (step representation). A lattice path in $\mathcal{L}(W, H)$ represented by λ in height representation has *step representation* $\underline{n}^\lambda = (n_0^\lambda, \dots, n_W^\lambda)$ where the component n_a^λ is the number of steps up that λ takes at $x = a$. Equivalently, it is the discrete difference of the height representation, written $\underline{n}^\lambda = \Delta\lambda$ with components

$$n_a^\lambda = \lambda_a - \lambda_{a-1},$$

see example in Figure 2 in blue. By convention, $\lambda_{-1} = 0$. The step representation is suggestively denoted by $\underline{n}$ to indicate the connection to bosonic Fock states, and identified by a superscript whenever required.

Both representations are clearly equivalent, and to go from step representation $\underline{n}$ to height representation $\lambda^\underline{n}$ (note the superscript to identify the path), we take the summation:

$$(\lambda^\underline{n})_a = \sum_{b=0}^a n_b. \quad (2.3)$$

2.3.2 Fock states represented by lattice paths

In Definition 2.5, we define the step representation of a lattice path, suggestively denoting it

$\underline{n}$ highlighting the connection to Fock (number) states. We show two example basis states in the right path of Figure 1c. We now make this connection formal:

Lemma 2.6 (bosonic states are lattice paths). The bosonic Fock space $\mathcal{F}^n \mathbb{C}^m$ of n indistinguishable photons in m modes is spanned by lattice paths in a rectangle of size $(m-1, n)$:

$$\mathcal{F}^n \mathbb{C}^m = \langle \mathcal{L}(m-1, n) \rangle_{\mathbb{C}}, \quad (2.4)$$

where we use the notation $\langle S \rangle_{\mathbb{C}}$ (or just $\langle S \rangle$) for a $\mathbb{C}$ -vector space spanned by elements of a set S seen as an orthogonal basis.

Proof. We show that the dimension is correct. A lattice path in $\mathcal{L}(W, H)$ must start at the origin and take H steps up and W steps right to reach (W, H) . It takes a total of $W + H$ steps, and any order of the steps is allowed, so the number of paths is the combination number $\binom{W+H}{H}$. Setting $W = m-1$ and $H = n$:

$$|\mathcal{L}(m-1, n)| = \binom{m-1+n}{n} \stackrel{(2.1)}{=} \dim \mathcal{F}^n \mathbb{C}^m.$$

Finally, note that all lattice paths end at height n , so each path in step representation, which contains the sizes of the steps up, corresponds to distributing the n photons into m modes. We conclude that lattice paths in $\mathcal{L}(m-1, n)$ indeed correspond to basis states of $\mathcal{F}^n \mathbb{C}^m$. ■

So far, we have a convenient way to visually represent number states that implements the constraint that there are in total n photons distributed between m modes. In the next section, we show that lattice paths can help us implement more constraints on the state space.

2.3.3 Loops give downsets

The circuit topology of a loop-based system allows incoming photons to end up only in some modes. We can capture this using a region in the lattice that represents the subspace of the valid number states. As shown in Figure 3, the important new information encoded by this region are the constraints on the maximal number of photons in a given mode, which form a nondecreasing sequence that defines a lattice path, and all paths below it are allowed. This is enough to

describe the state space because photons are indistinguishable and we only know how many are in each mode.

We show how this works for a system with length $\ell_j = 1$ for each loop, which is one of the main results of [10]. Our work in Section 4 extends this to more general systems.

We take the simplest example using a system of a single loop $X = L_1$ acting on an input state $|1^5\rangle$ on 5 modes in Figure 3. At each of the output wires of the circuit, we write down the set of possible numbers of photons measured there: these have the form $\llbracket 0, \mu_a \rrbracket = \{0, 1, \dots, \mu_a\}$, where μ_a denotes the maximum number of photons that could exit in mode a . Intuitively, μ_a can be found by following the possible paths of the photons through the circuit X . More formally, if the input state is $|\underline{n}\rangle$, we can express the maximum number of photons in output mode a as

$$\mu_a = \sum_b n_b,$$

where the summation runs over the input modes b from which photons can reach output mode a .

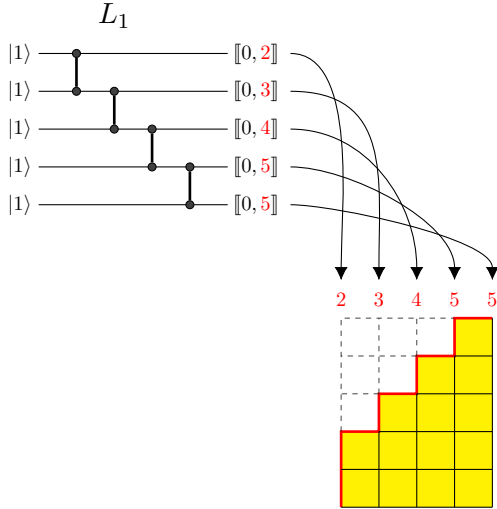


Figure 3: An example showing how we obtain a maximal lattice path μ from a given circuit and input state, in this case L_1 and $|1^5\rangle$. On the output modes are sets of possible photon numbers measured there, and we show how these define the lattice path μ (red). The possible measurement outcomes are all the lattice paths lower than μ – these are the paths constrained to the yellow-shaded region under μ .

Due to the staircase structure of L_1 , the sequence $\mu = (\mu_0, \dots, \mu_{m-1})$ is nondecreasing and terminates in n , the total number of photons.

This means it corresponds to a height representation of a lattice path in $\mathcal{L}(m-1, n)$, shown in red in the example of Figure 3. The set of possible output basis states is exactly the set of paths lower than μ , i.e. those constrained to the yellow-shaded region in the figure. We refer to μ as the *maximal lattice path*, and we call the set of paths below or equal to it the *downset* $\mu\downarrow$. This exactly describes the state space which has the following form:

Definition 2.7 (cumulative space). Let $\mathcal{K}$ be a subspace of $\mathcal{F}^n \mathbb{C}^m$. If there exists a maximal lattice path $\mu \in \mathcal{L}(m-1, n)$, corresponding to a basis state $|\underline{n}^\mu\rangle$ in $\mathcal{F}^n \mathbb{C}^m$, such that $\mathcal{K} = \langle \mu\downarrow \rangle_{\mathbb{C}}$, then we call $\mathcal{K}$ a *cumulative space*.

2.3.4 Interpretation

The reason to make the above definition and call it a cumulative space is that the loop L_1 acts as an *accumulator* of photons. It collects incoming photons, and may release them into any subsequent time-bin. Notice that the maximal path μ corresponds to photons leaving the system as soon as possible, i.e. never entering the loop.³ The paths below μ correspond to some photons entering the loop and leaving later than their starting time-bin, with the extreme being the bottom path $\lambda^\perp = (0, \dots, 0, n)$ where all photons stay in the loop until the last time-bin mode.

Finally, we shortly note that in this context of lattice paths, *lower than* has a precise meaning in terms of a partial order on the set $\mathcal{L}(m-1, n)$, called the *Young order*. Two paths λ and μ satisfy $\lambda \leq \mu$ iff $\lambda_a \leq \mu_a$ for each component a . The heights of λ are lower in the lattice than those of μ , so indeed λ belongs to the region below μ in Figure 3, which is the downset

$$\mu\downarrow = \{\lambda \in \mathcal{L}(m-1, n) : \lambda \leq \mu\}.$$

In the main body, we give a mostly non-order-theoretic presentation, and instead we only elaborate on the formal details in Appendix B.

³Except of course the first photon, which is preloaded into the loop by convention (see Section 2.2.1). This photon also immediately exits.

3 Loop-based structure allows simulation of certain interferometers

We present a new wavefunction evolution algorithm to classically simulate loop-based interferometers exploiting their staircase structure by performing partial measurements when modes become unused.

3.1 Progressive simulation algorithm

Observe in Figure 1a that due to the staircase shape of the circuits corresponding to loops, each mode has a time in the circuit after which it is never affected by any subsequent operation. Our algorithm is a hybrid between strong and weak simulation: it evolves the wavefunction through small components of the circuit, and after each of them, it simulates the measurement of a single mode that is never needed again, collapsing the wavefunction to a smaller subspace in the process. Due to the collapse, the memory required to store the state vector stays more manageable. Furthermore, we show that the strategy of measuring single modes and collapsing the state means that we only strongly simulate circuits with a low effective number of modes. Therefore our algorithm can tackle even large loop-based systems that could not be simulated otherwise.

3.1.1 Decomposition of the circuit

We describe the idea using the simple system $L_1 \circ L_3$ shown in Figure 1a. We push the measurement of mode 0, and all beamsplitters that influence it, to the left. At the same time, we push all subsequent beam splitters to the right of the measurement, obtaining Figure 1b. Two beamsplitters can be pushed past each other whenever they do not act on the same modes. Note that moving measurements to the left, we are identifying their causal cones, as shown in the more complex example of Figure 4 where we highlight the paths in the circuit through which photons could arrive to output mode 0.

In Figure 4, notice that to find all the modes and beamsplitters relevant to output 0, we just need to find the farthest reachable mode. We follow the path with arrows ($\leftarrow$) from output 0 backward, and use each beamsplitter to jump to a later time-bin mode until we reach the first loop of length 1. Since this loop corresponds to a

staircase of nearest-neighbour beamsplitters, all the time-bin modes identified by lower numbers are also connected to output 0 (shown by yellow fill).

We iterate the procedure of moving measurements left for all output modes in the order $0, 1, \dots$, transforming the example circuit in Figure 1a to the form of Figure 1b which we call the *progressive decomposition* to components $(P_a)_{a=0}^{m-1}$. Each component P_a contains only the beamsplitters inside the causal cone of mode a that have not appeared in the previous components $P_0, \dots, P_{a-1}$. This means that even though the set of beamsplitters relevant to an output mode a includes some of the beamsplitters relevant for mode $a - 1$, the component P_a contains only the “new” beamsplitters.

After rewriting the circuit as above, observe the effective number of modes in all the components P_a is bounded by a number which only depends on the loop lengths. In Figure 1b, this is the maximum height of a box P_a .

Theorem 3.1 (relevant modes). In a progressive decomposition of a loop-based system X made of loops of lengths $\underline{\ell} = (\ell_1, \dots, \ell_\Lambda)$, each component P_a acts non-trivially on at most the number of *relevant* modes

$$R = 1 + \sum_{i=1}^{\Lambda} \ell_i. \quad (3.1)$$

This follows from the intuition above; we also provide a formal proof in Appendix A.

3.1.2 Simulation

We can understand the progressive decomposition as a sequence of smaller circuits P_a , each of which only acts on at most R modes, and only includes a few beamsplitters. We evolve the state vector through only these beamsplitters, and then we perform a partial measurement of the top mode a of the component, collapsing the wavefunction to a subspace compatible with the measurement outcome. The collapsed wavefunction becomes the input state of the next component. We iterate the procedure for all components, thus progressively generating the full sample $|n'\rangle$. We summarize the algorithm, labelled Algorithm PS, as pseudocode in Figure 5.

Observe in Line 1 in the pseudocode that we call the strong simulation as a subroutine. This

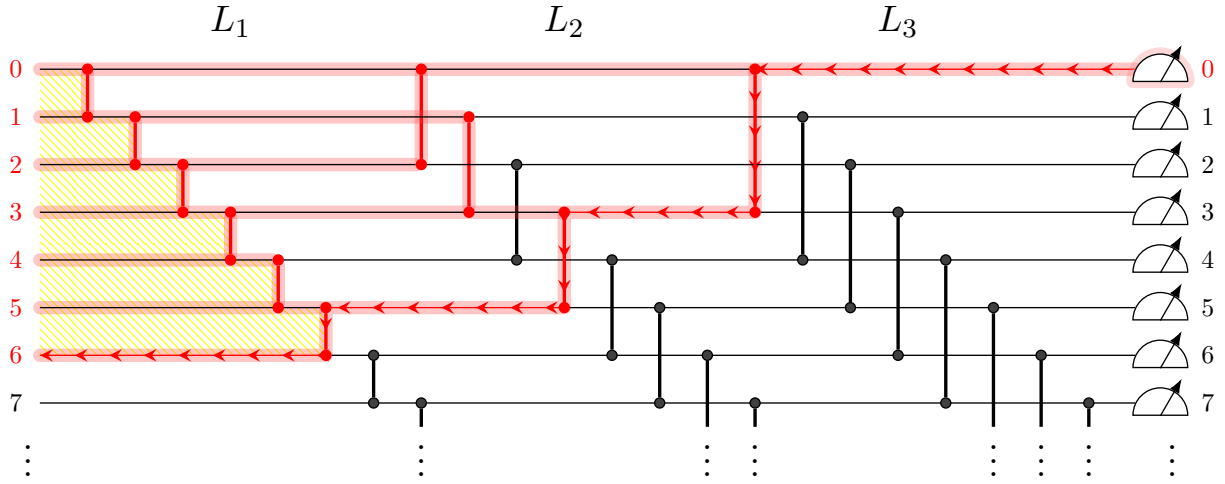


Figure 4: A view of the input modes and beamsplitters influencing output mode 0 in a system $L_1 \circ L_2 \circ L_3$. We highlight in red the wires along which photons may reach output mode 0 on the right. The path reaching the maximum input mode is further emphasized with arrows ($\leftarrow\leftarrow\leftarrow$). The relevant beamsplitters are drawn in red as well – these are the beamsplitters that form the component P_0 . Observe that the first loop L_1 is a staircase of nearest-neighbour beamsplitters, so if a mode a is relevant, so are the modes $< a$; this fact is emphasized in the yellow area.

Algorithm PS: Progressive simulation

Input: Number of modes m , vector of loop lengths $\underline{\ell} = (\ell_1, \dots, \ell_\Lambda)$, vector of beamsplitter angles $(\theta_i)_i$, input state $\underline{n} = (n_0, \dots, n_{m-1})$.

Output: A single sample

$$\underline{n}' = (n'_0, \dots, n'_{m-1}).$$

- 1 $(P_a)_{a=0}^{m-1} \leftarrow$ progressive decomposition of $L_1 \circ \dots \circ L_\Lambda$ obtained by pushing measurements to the left and reordering beamsplitters (together with their angles $(\theta_i)_i$) accordingly
 - 2 $|\psi\rangle \leftarrow |\underline{n}\rangle$
 - 3 **for** $a = 0, \dots, m-1$ **do**
 - 4 $|\psi\rangle \leftarrow$ evolve $|\psi\rangle$ through component P_a as a subroutine
 - 5 $p(n'_a) \leftarrow$ compute the marginal probability of measuring n'_a photons in output mode a , for each possible n'_a
 - 6 $n'_a \leftarrow$ random sample from the distribution p
 - 7 $|\psi\rangle \leftarrow$ collapse $|\psi\rangle$ to subspace compatible with measurement n'_a
 - 8 **return** $\underline{n}' = (n'_0, \dots, n'_{m-1})$
-

Figure 5: Pseudocode of the progressive simulation algorithm. This algorithm is independent of how the state is stored (e.g. as a dictionary).

gives us the freedom to choose any wavefunction simulation algorithm, as long as it can accept a

superposition state as input. This is required because the projected state after measuring a mode a , which is the input to P_{a+1} , is generally a superposition of several number states.

3.2 Evaluating the algorithm

To evaluate how useful the progressive algorithm is, we aim to estimate the cost of using it as a function of different parameters of the system to be simulated.

We specifically focus on the memory complexity because this gives us a lower bound on the running time for any choice of exact wavefunction simulator. This is the case because any wavefunction simulator must at least spend time to read the entire stored state. Focusing on memory is further justified by the fact that the current best algorithm `SLOS_full` performs wavefunction simulation in time linearly bounded by memory [14].

The three main contributions to memory complexity come from the following:

1. The effective number of modes in each sub-circuit P_a , given by the value R from Theorem 3.1.
2. The total number of photons in the system.
3. The possible positions of photons, given by the input state and the circuit topology.

Point number 3 is the focus of the next Section 4 and one of the main technical contributions of the present paper. However, already using the

first two points, we can roughly estimate which architectures are easy to simulate. For example, we can clearly hold a billion coordinates in the memory of a high-end computer, so we set the dimension to 10^9 . Assume, as an upper bound, that we need the full Fock space with dimension in eq. (2.1). We need to simulate R modes, and for simplicity assume we have R photons to maximize the binomial in (2.1). Then R should satisfy

$$\binom{2R-1}{R} \approx 10^9,$$

which is the case when $R \approx 16$. This implies we can easily simulate systems made of loops of lengths, for example, $\ell = (1, 14)$, $(1, 5, 9)$, or $(1, 2, 4, 8)$.

However, this method is a very coarse approximation. Not only may there be fewer or more photons in a component P_a to be simulated, invalidating the assumption that $n \approx R$ above, but the circuit topology restricts where those photons can be. This means the actual state space is generally very different. We present a more granular and robust way to estimate complexity in the following sections.

4 A new formalism to enumerate the state space

To quantify the complexity of our progressive simulation algorithm, in Section 4.1 we expand the formalism of Brádler & Wallner [10] from loops of length one, presented in Section 2.3, to more general systems where loops have different lengths. Then in Section 4.2, we show how to encode the outcomes of mid-circuit partial measurements. Together, these allow us to describe the state space throughout simulation, an important tool in computing the complexity in Section 5.

4.1 Evolution through long loops

Recall from Section 2.3.3 that we describe the possible outputs of a loop-based system by a sequence $\mu = (\mu_0, \dots, \mu_{m-1})$ where each μ_a is the maximum number of photons that can be measured in output mode a . For a system of loops of length one, Brádler and Wallner [10] show that μ defines a lattice path in height representation, and that the state space is fully characterized by the set of lattice paths lower than μ . In order for

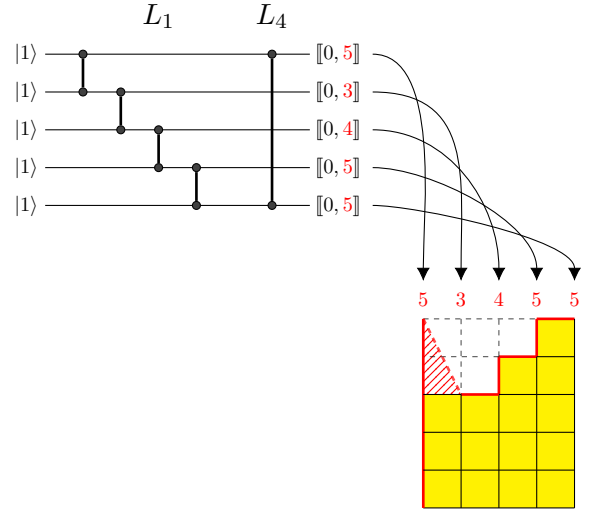


Figure 6: Example circuit that cannot be described by the original lattice path formalism of [10]. The maximum numbers of photons form a sequence $\mu = (5, 3, 4, 5, 5)$ which is not nondecreasing, and hence not a lattice path in height representation. As a lattice path, μ would have to descend (hatched part in the picture) and this is not allowed.

μ to be a valid lattice path, it must be a nondecreasing sequence.

This breaks when we allow a loop of longer length. See for example in Figure 6 that the loop L_4 allows the exchange of photons between the top and bottom mode, leading to a maximal “path” $\mu = (5, 3, 4, 5, 5)$. As μ is not nondecreasing, it is not in fact a valid lattice path. We show that this issue can be addressed by introducing mode permutations into this formalism.

4.1.1 Lattice paths with mode permutations

To encode a loop of length $\ell_j > 1$, we introduce mode permutations in the lattice path representation so that they appear in an order of nondecreasing maximum photon number. Recall that in Section 2.3.3, the nondecreasing property of the maximal path μ arises due to a loop L_1 accumulating incoming photons, and releasing them in any subsequent time-bin modes. Individual lattice paths in $\mu_\downarrow$, via their step representation, capture ways to distribute the fixed number of photons between modes. However, the order of modes in the lattice path representation has no physical meaning, serving only as labels that can be reordered.

In the following, we show that even with modes permuted, we can still fully describe the state

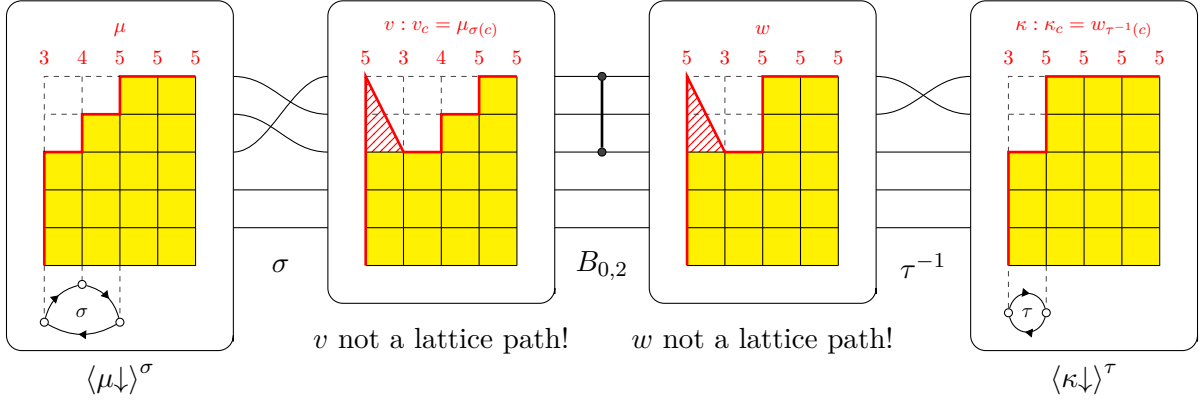


Figure 7: An example of the PCS evolution rule from Theorem 4.2, following the same steps as the accompanying text. We start with a PCS $\langle \mu \downarrow \rangle^\sigma$ obtained by the circuit and input from Figure 6. On the left, we display the maximum path μ and the permutation σ acting on the PCS. Next, we apply the permutation σ to the top path to obtain the vector v describing the maximum photon numbers in real (unpermuted) modes, and we highlight that this is not a lattice path. Then we apply a beamsplitter between modes 0 and 2, obtaining the new vector of maximum photon numbers w , which is again not a lattice path. Finally, we sort the modes using a permutation τ^{-1} to obtain a new maximum lattice path κ , landing in a new PCS $\langle \kappa \downarrow \rangle^\tau$. Note that by construction the permutation acts to permute modes from PCS to the physical space, and hence sorting the modes from physical to PCS in the last step is done by τ^{-1} .

space by a downset $\mu \downarrow$. To do this, we generalize the notion of cumulative space from Definition 2.7:

Definition 4.1 (PCS). A subspace $\mathcal{H} \subseteq \mathcal{F}^n \mathbb{C}^m$ is called a *permuted cumulative space (PCS)* if there exists a cumulative space $\langle \mu \downarrow \rangle \subseteq \mathcal{F}^n \mathbb{C}^m$ and a permutation of modes $\sigma \in S_m$ that acts as an isomorphism from $\langle \mu \downarrow \rangle$ to $\mathcal{H}$ sending

$$|n_0, \dots, n_{m-1}\rangle \mapsto |n_{\sigma(0)}, \dots, n_{\sigma(m-1)}\rangle.$$

We write $\mathcal{H} = \langle \mu \downarrow \rangle^\sigma$.

4.1.2 Non-nearest-neighbor beamsplitter

To show how to use permuted spaces to reason about general loop-based systems, we focus first on describing the action of a basic building block, a single beamsplitter between arbitrary modes a and b that are not necessarily nearest neighbors. In Figure 7, we show an example of the same steps that follow in the explanation.

We define a vector v where $v_c = \mu_{\sigma(c)}$ is the maximum possible number of photons in mode c . Due to the cumulative nature of the input state space, the maximum number of photons in either modes a and b after applying the beamsplitter is $\max\{v_a, v_b\}$. We write a new vector of maximum

photon numbers w_c with components:

$$w_c = \begin{cases} \max\{v_a, v_b\} & \text{if } c \in \{a, b\}, \\ v_c & \text{if } c \notin \{a, b\}. \end{cases}$$

Since a and b are not necessarily nearest neighbors, and because v may not be nondecreasing in the first place, it is possible that w is not a nondecreasing sequence. Hence w is not a valid lattice path. We sort it in nondecreasing order and thus permute the modes. We call the permuted sequence κ which is a valid lattice path in height representation, and we call τ the inverse of the permutation we used (where the permutation maps $\kappa \mapsto w$).

We summarize the procedure as the following evolution rule:

Theorem 4.2 (PCS evolution). Suppose $\mathcal{H} = \langle \mu \downarrow \rangle^\sigma$ is a PCS with a mode permutation σ applied. Then applying a beamsplitter between arbitrary modes a, b maps $\mathcal{H}$ to a PCS $\langle \kappa \downarrow \rangle^\tau$ with a new mode permutation τ , where

$$\kappa_{\tau(c)} = \begin{cases} \max\{\mu_{\sigma(a)}, \mu_{\sigma(b)}\} & \text{if } c \in \{a, b\}, \\ \mu_{\sigma(c)} & \text{if } c \notin \{a, b\}; \end{cases} \quad (4.1)$$

and where τ assures that κ is non-decreasing.

Proof. This follows from the reasoning above the theorem statement. ■

It follows that to obtain the state space at the output of a loop-based system, we apply the above rule iteratively. This simplifies to the following:

Corollary 4.3. Suppose X is a loop-based system made of loops of lengths $\underline{\ell} = (1, \ell_2, \dots, \ell_\Lambda)$, where $\ell_1 = 1$ and other lengths are arbitrary. Then there exists a lattice path in height representation μ and a mode permutation $\sigma \in S_m$ such that the space of possible output Fock states resulting from applying X to the input $|\underline{n}\rangle$ is a PCS $\langle \mu \downarrow \rangle^\sigma$ with

$$\mu_{\sigma(a)} = \sum_b n_b$$

where the summation runs over input modes b that are reachable from output mode a (see Section 3.1.1). The permutation σ ensures μ is non-decreasing.

4.1.3 Discussion

The tools above allow us to fully and efficiently describe the output state space of any system of loops where $\ell_1 = 1$, on any input basis state, and any number of modes. This includes even large systems where the state space would be too big for a computer memory, since in our formalism, we need only store two objects: μ and σ . The condition that $\ell_1 = 1$ stems from the fact that our evolution rule in Theorem 4.2 requires its input space to be a PCS: from an arbitrary input (not a PCS), in particular a single number basis state, we obtain a PCS by applying a loop L_1 (see Section 2.3.3). However, requiring that $\ell_1 = 1$ is reasonable in the context of time-bin loop-based systems, and it is always the case in the present paper.

We use the lattice path formalism to count the dimension of the state space characterized by a downset. Obtaining a general closed-form formula that counts any downset is difficult, though possible when the loop lengths are restricted (e.g. Brádler & Wallner proved a closed-form formula when all lengths are $\ell_j = 1$ [10]). However, in Appendix B.4, we provide a dynamic programming algorithm that can efficiently count lattice paths in general systems without such restriction.

The applicability of Theorem 4.2 goes beyond staircase-shaped circuits, as it allows us to characterize the evolution of the state space beamsplitter-by-beamsplitter, particularly in the order given by the progressive decomposition (see

Section 3.1.1). The last ingredient missing to fully describe the state space throughout all of the steps of the progressive algorithm are partial mid-circuit measurements, which we develop in the next section.

4.2 Measuring arbitrary modes

In the progressive simulation algorithm, after evolving the statevector through a component P_a of the progressive decomposition, we perform a measurement of its top mode, after which it is removed from the simulation. To fully describe the state space at every step during the simulation, we need to capture this partial measurement in our lattice path formalism. Even though we always measure the top mode, due to the permutation of a PCS this may be represented by a different mode in the formalism. We thus adapt the formalism so that it can encode the measurement of an arbitrary mode of a cumulative space.

Our key contribution in this section is showing that a state space described by lattice paths can still be described by lattice paths after a measurement is performed. Specifically, we show the measurement of x photons in mode a sends a maximal path μ over m modes and n photons to the new maximal path $\mu^{(n_a=x)}$ over $m-1$ modes and $n-x$ photons.

We follow three steps, shown in a running example in Figure 8: First, we encode the projection to a subspace where each basis state has $n_a = x$ (Section 4.2.1). This is not a single downset anymore, but it is made of a collection of regions in the lattice, each defined by λ_{a-1} , the number of photons in modes $\llbracket 0, a-1 \rrbracket$. Second, we delete the measured mode a and the x photons inside by squashing the lattice diagram (Section 4.2.2) which is vital for merging the aforementioned regions back together. The final step is the merge itself: we show that taken together, the squashed regions give a new PCS, and hence our efficient representation can encode measurements (Section 4.2.3).

4.2.1 Projection

When we measure x photons in mode a , we project the statevector to the subspace that is compatible with that measurement. This is the step (a) $\rightarrow$ (b) in the running example in Fig. 8. We only keep those lattice paths that have a step

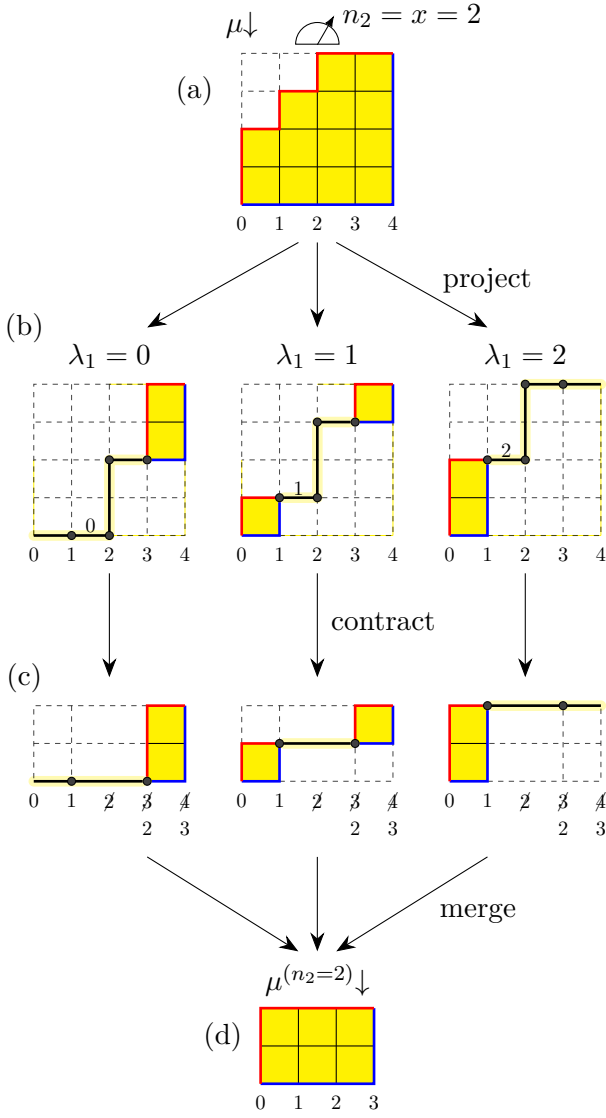


Figure 8: Encoding a measurement outcome in the lattice path formalism. **(a)** State space before a measurement which is a cumulative space $\langle \mu \downarrow \rangle$ with $\mu = (2, 3, 4, 4, 4)$. We measure $x = 2$ photons in mode $a = 2$. **(b)** First, we *project* to a subspace compatible with the measurement, keeping only those lattice paths λ that have $n_a^\lambda = x$. The set of these paths is, in general, not a single skew diagram in the lattice, but a union of several skew diagrams $\{\mu^q/\nu^q\}_q$ where $q = 0, \dots, q_{\max}$. Each μ^q/ν^q represents the subspace with q photons in modes $< a$. These constraints give μ^q/ν^q the S-shaped region centered at mode a , starting at height q , and with step x . **(c)** Next, we delete the measured mode and the photons within. We implement this as a *contraction* of the S-shaped region: squash the diagram vertically to ensure that at mode a , the step has size zero. Then erase the vertical line that corresponds to mode a , merging the boxes to the left and to the right of it. Relabel the modes to the right of a as $b \mapsto b - 1$. **(d)** Finally, we *merge* the contracted diagrams. This results in a contiguous region in the lattice, the downset $\mu^{(n_a=x)} \downarrow$.

of size x in mode a :

$$\{\lambda \in \mu \downarrow : n_a^\lambda = x\}. \quad (4.2)$$

The above is generally not the set of paths in some downset, or even a contiguous region in the lattice: the projection removes certain paths, creating gaps in the region. This is a problem for our representation, since this would no longer allow us to efficiently characterize very large state spaces using a single maximal path μ .

The solution is to decompose (and later recompose) the above set (4.2) into lattice regions defined by the number of photons in modes $\llbracket 0, a-1 \rrbracket$, encoded by restricting the paths λ to have height $\lambda_{a-1} = q$ (see (2.3)), for each possible choice of q . These regions, depicted in Figure 8b, correspond to a collection of disjoint subspaces of the state space, and are represented by regions of the lattice that can be efficiently described by their boundaries, i.e. the minimum and maximum paths. We call them *skew diagrams* μ^q/ν^q where q is the choice of λ_{a-1} . Where a downset $\mu \downarrow$ includes all lattice paths in the region below μ , a skew diagram μ^q/ν^q contains paths between and including μ^q and ν^q :

$$\mu^q/\nu^q = \{\lambda : \nu^q \leq \lambda \leq \mu^q\}. \quad (4.3)$$

In Figure 8, we draw μ^q red and ν^q blue wherever they differ. Notice that they all include an S-shape that implements the step of size x in mode a starting at height q . The value of q ranges between 0 and $q_{\max} = \min\{\mu_{a-1}, \mu_a - x\}$, the latter easily found as the highest y coordinate in the diagram where the start of the S-shape of height x can be placed within a diagram $\mu \downarrow$.

Informally, the systematic way to obtain all μ^q/ν^q is to start with $q = 0$, place the S-shape at the corresponding position in the lattice, and restrict the yellow region to be compatible with the S-shape. Then increase q , sliding the S-shape upward, and repeat the procedure until we reach the top of the lattice ($q = q_{\max}$). We show the formal details of computing all skew diagrams μ^q/ν^q in Appendix B.2.1.

4.2.2 Contraction

The next step is to delete the measured mode a and the photons it contained from all the skew diagrams produced in the previous step. To do this, we remove the x measured photons by removing

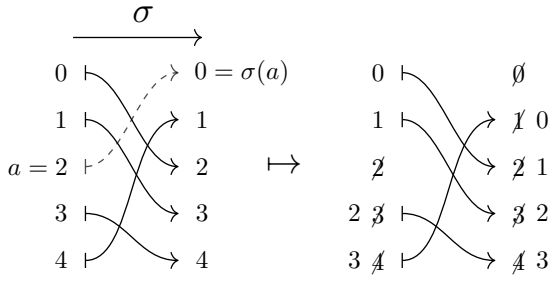


Figure 9: Example action of the contraction of mode $a = 2$ on the permutation σ of a PCS. On the LHS, we delete input mode a on the left, as well as its target, the output mode $\sigma(a)$ on the right, related by the dashed arrow. All the modes below these, on each side, have their labels shifted by -1 . We obtain the contracted permutation on the RHS.

x rows from each diagram (i.e. remove boxes between vertical positions q and $q+x$), then remove one column at position a . This implements a contraction of the S-shaped region in μ^q/ν^q which is shown as the step (b) $\rightarrow$ (c) in the running example in Figure 8. We are left with contracted skew diagrams that represent subspaces of a system of $m-1$ modes and $n-x$ photons. We provide a formal presentation in Appendix B.2.2.

If the state space is a PCS with a nontrivial permutation σ , we implement the contraction on σ by deleting the input a , output $\sigma(a)$, and shifting the labels of modes following the deleted modes by -1 . We show an example in Figure 9.

4.2.3 Merge

Finally, we merge all of the contracted skew diagrams, one for each $q = 0, \dots, q_{\max}$, to again obtain a representation where a single lattice path characterizes the whole state space as a downset. We show this in the step (c) $\rightarrow$ (d) of the running example in Figure 8.

Informally, to see why this yields a downset, consider the following: After contraction of the S-shapes, each skew diagram encodes only the choice of $\lambda_{a-1} = q$, that is, it enforces that the paths of that diagram pass the point $(a-1, q)$. However, the paths are otherwise free in the other modes, as long as they are consistent with the former constraint, as well as the original constraints imposed by μ/ν ; see Figure 8c. We can imagine “scanning” the contracted lattice, setting $q = 0$, and writing down all the paths that satisfy the constraint $\lambda_{a-1} = q$, then advancing to $q = 1$, etc. Note in particular that we have a skew dia-

gram for each $q \in \llbracket 0, q_{\max} \rrbracket$ without gaps. This captures all the valid lattice paths after a measurement, as seen in the example in Figure 8d. However, this part is formally nontrivial, and we provide more detail in Appendix B.2.3.

4.2.4 Full description of measurement

Putting the three steps together, we obtain one of our main results. It gives us the ability to efficiently encode the effect of a measurement of an arbitrary mode, with a given measurement outcome, on the state space using the lattice path formalism:

Theorem 4.4 (measurement). Starting with a cumulative space $\langle \mu \downarrow \rangle$, the state space left after measuring x photons in mode a is a cumulative space $\langle \mu^{(n_a=x)} \downarrow \rangle$ where the maximal path $\mu^{(n_a=x)}$ is the contraction of $\mu^{q_{\max}}$. The latter is the maximum path of the skew diagram corresponding to $q = q_{\max} = \min\{\mu_{a-1}, \mu_a - x\}$.

The proof can be found in Appendix B.2.3.

4.3 Tracking the state space in progressive simulation

Together, the maximal lattice path μ^t and the mode permutation σ^t at step t fully describe the state space throughout the simulation, where each step is either an individual beamsplitter or a measurement. If step t is a beamsplitter, then we obtain μ^{t+1} and σ^{t+1} using Theorem 4.2. Note that if the beamsplitter is part of the initial L_1 , as described in Appendix B.3, we must also append a new mode to the PCS. On the other hand, if step t is a measurement, we choose its outcome number of photons, and obtain μ^{t+1} using Theorem 4.4 and σ^{t+1} as described in Section 4.2.2.

Our extension of the lattice path formalism allows us to determine the size of the state space that the progressive simulator must keep in memory to reach a specific output measurement $|\underline{n}'\rangle$. However, this does not tell us how $|\underline{n}'\rangle$ should be chosen. To quantify metrics such as the average-case memory complexity of simulating a given architecture, we would need to sample $|\underline{n}'\rangle$ from a generally hard boson sampling distribution. That would be counterproductive because we want to estimate the complexity of sampling from this distribution in the first place. In Section 5, we show how to solve this issue and thus use the lattice

path formalism to quantify the complexity of the progressive algorithm itself.

5 Quantifying the simulation complexity of loop-based interferometers

Now that we have a tool to characterize the state space at every point in our simulation, including individual beamsplitters as well as measurements, we are almost ready to use this to estimate the complexity of simulating loop-based systems, defined in terms of the memory requirement:

Definition 5.1 (memory complexity). Given a loop-based system with loop lengths ℓ , total number of modes m , and an input state $|\underline{n}\rangle$, we define a function M that captures the memory required to run a simulation of that system producing the output pattern $\underline{n}'$:

$$M(\underline{n}') = \max_t |\mu^t \downarrow|$$

where μ^t is the maximal lattice path at step t of the simulation.

This metric captures the maximum memory needed to store the state vector over the course of the simulation. We note that M does not assume a choice of strong simulator or how the data is represented: it is simply the number of nonzero coordinates to be stored. Since the number of photons n'_a measured in mode a at any given time affects the size of the state vector after the measurement, the value of M does depend on the choice of $\underline{n}'$.

As this expression is defined in terms of a single output $\underline{n}'$, to quantify the complexity of simulating an experiment, we must estimate the statistics of $M(\underline{n}')$ over a range of possible values of $\underline{n}'$. One useful metric is the average complexity of an experiment:

$$\bar{M} := \mathbb{E}_{\underline{n}' \sim p} [M(\underline{n}')] = \sum_{\underline{n}'} p(\underline{n}') M(\underline{n}'), \quad (5.1)$$

This is the expectation over all possible samples $\underline{n}'$ taken with respect to the physical probability distribution p , i.e. the actual outcome probability.

The distribution p is hard to compute or sample from for all but the smallest systems, so we need a tractable way to evaluate the expectation (5.1). To solve this issue, in Section 5.1, we build a heuristic probability distribution that is easy to sample from and can be used to predict the

memory complexity. We validate it by comparison against samples from small and medium-size architectures that can be simulate. Then in Sections 5.2 and 5.3, we use the heuristic to study the behavior of complexity with respect the total number of modes and the lengths of loops in large systems in order to assess their potential for quantum advantage.

5.1 A good heuristic probability distribution

To evaluate the expectation $\bar{M}$ in Eq. (5.1) without having to sample from the true distribution p for a large system, we propose instead using a heuristic probability distribution p_H that is easy to sample from and provides a good estimate of $\bar{M}$. Given the same input state and loop structure that gave rise to p , we now assume that all possible outcomes $\underline{n}'$ are equally likely, thus defining p_H as the uniform probability:

$$p_H(\underline{n}') = \frac{1}{\text{number of possible outcomes}}.$$

There is a simple and efficient way to generate samples $\underline{n}'$ from the distribution p_H mode-by-mode using the progressive decomposition:

Lemma 5.2. Suppose μ is a maximal path after simulating the circuit up to the end of component P_a , just before the measurement of mode a . In particular, μ already encodes the previously obtained measurement outcomes $n'_0, \dots, n'_{a-1}$. Then the marginal heuristic probability of measuring x photons in mode a is

$$p_H(n'_a = x \mid n'_0, \dots, n'_{a-1}) = \frac{|\mu^{(n_a=x)} \downarrow|}{|\mu \downarrow|}, \quad (5.2)$$

where previous measurements $n'_0, \dots, n'_{a-1}$ are fixed, the value $x \in \llbracket 0, \mu_a \rrbracket$, and $\mu^{(n_a=x)}$ is the maximal lattice path after the measurement.

Sketch proof. The maximal path μ already encodes the outcomes $n'_0, \dots, n'_{a-1}$ and the probability of outcomes is uniform, so we just need to count the paths in $\mu \downarrow$ with $n_a = x$. ■

Note that the uniform heuristic p_H is very different from the true SPBS distribution p , but is a good first choice because it requires no information about p . We show in Section 5.1.1 that p_H does in fact provide a good estimate of the statistics of the memory complexity, giving similar results to the real distribution.

Comparison of complexities estimated using heuristics to real complexities

Distribution of memory per sample M (log scale)

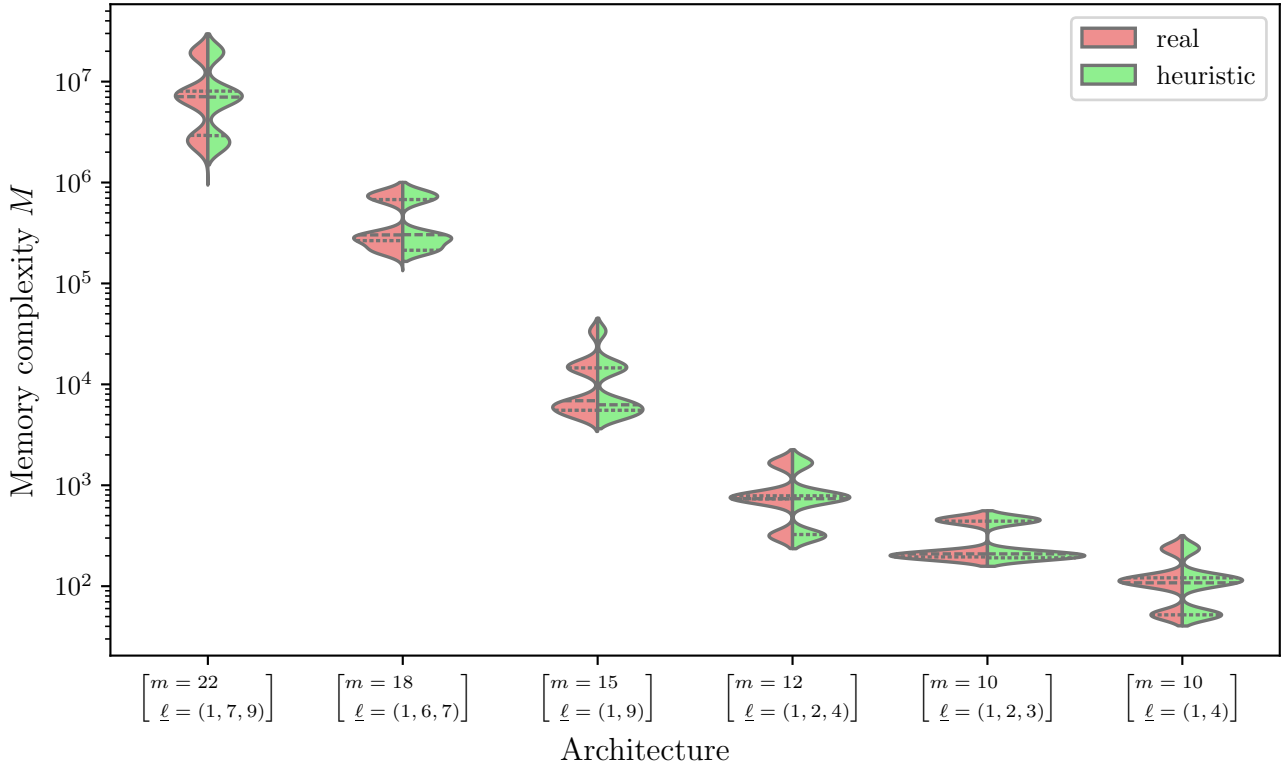


Figure 10: Validation of the heuristic probability distribution p_H . The horizontal axis shows different architectures $(m, \underline{\ell}, |\underline{n}\rangle)$ where $|\underline{n}\rangle = |1, 0, 1, 0, \dots\rangle$ with $\lceil \frac{m}{2} \rceil$ photons. The vertical axis shows the memory requirement to simulate samples from a given architecture using our method, and is scaled logarithmically. The data are shown as mirrored violin plots, showing the distribution of memory with respect to true probability $\bar{p}$ in red on the left, and with respect to the heuristic p_H in green on the right. Inscribed in the violins are the quartiles: median dashed, 25-th and 75-th percentiles dotted.

5.1.1 Validation of the heuristic

We empirically validate on small, easy-to-simulate circuits that the heuristic p_H can be used to compute the average memory complexity. Since a real boson sampling distribution p depends on the choice of beamsplitter parameters $(\theta_i)_i$, in our numerical experiments we average over several choices of $(\theta_i)_i$ drawn uniformly at random. Our validation consists of showing the following approximation:

$$\mathbb{E}_{\underline{n}' \sim p_H} [M(\underline{n}')] \approx \mathbb{E}_{\substack{(\theta_i)_i \\ \underline{n}' \sim p}} [M(\underline{n}')] \quad (5.3)$$

In the following, we call $\bar{p}$ the probability distribution on the RHS above which averages over $(\theta_i)_i$.

We choose, here and in the following sections, the input state $|\underline{n}\rangle = |1, 0, 1, 0, \dots\rangle$ with $\lceil \frac{m}{2} \rceil$ photons. This choice is motivated by two considerations: First, current experimental efforts operate

roughly in this regime, for example [21] with input similar to ours, or [22] with a slightly different ratio between the numbers of photons and modes. This regime is interesting in near-term experiments because there is a large space of possible measurement outcomes even without perfect photon number resolution. Second, recent theoretical research supports the investigation of this regime (and generally $m = \mathcal{O}(n)$), showing that it is hard to simulate classically [23].

As shown in Figure 10, we find over several choices of loop architectures that the distribution of memory complexities using heuristic p_H closely matches the distribution obtained using $\bar{p}$. For visual convenience, we only show a small selection of experiments represented along the horizontal axis. The vertical axis represents the memory complexity $M(\underline{n}')$, and is scaled logarithmically. We take $N = 1000$ samples from each the true distribution $\bar{p}$ and the heuristic p_H , and from these

samples we estimate the corresponding probability distributions of M using *kernel density estimation* (KDE).⁴ We plot these estimates as mirrored violin plots: true distribution on the left in red, and heuristic on the right in green. For further comparison, each violin is inscribed with the quartiles: dashed line for median and dotted lines for 25-th and 75-th percentiles.

The results shown in Figure 10 confirm that the uniform heuristic p_H can be used to give good predictions for the memory complexity. This means we can use it to reason about systems where generating samples using the real distribution is not possible.

5.2 Complexity of a large loop-based system

Now that we have the tools to estimate the memory complexity of simulating arbitrary loop-based systems with our progressive algorithm, we can use them to analyze the largest loop-based systems built to date. We focus on a system with loop lengths $\underline{\ell} = (1, 6, 36)$ used by Madsen *et al.* in their *Borealis* GBS experiment [6], and we study the complexity of this architecture in the context of SPBS as a function of the total number of modes m . To do this, we replace the original Gaussian input with $|\underline{n}\rangle = |1, 0, 1, 0, \dots\rangle$.

In Figure 11, we plot the memory complexity as a function of the total number of modes m between 2 and 284. For each value of m , we take $N = 1000$ samples of memory complexity using the heuristic probability p_H . We study the average memory $\bar{M}$ (full line in the figure), as well as the 95-th percentile of the samples of M (dashed line). The latter represents the memory required to obtain almost all possible samples and we use it as a proxy for quantum advantage: if our classical computer has more memory than the 95-th percentile, then we can successfully run the progressive simulator at least 95 % of the time.

In the figure, we draw a comparison between the memory complexity and the memory available to a state of the art supercomputer, shown as a horizontal gray line. Our chosen value of 10^{15} is inspired by the OLCF-5 (Frontier) system [25], taking a loose assumption that all memory within the system can be used to store the quantum state. This overestimates the real ca-

capacity, but is independent of any implementation details. For comparison with the values $M(\underline{n}')$, we assume each nonzero coordinate is stored as a double-precision complex float.

Observe that there are three different regions in terms of m where the memory complexity has different behaviour, and there are abrupt jumps between these. The regions correspond to values of m where different numbers of loops are active. For example, if $m \leq 6$, then $L_1 \S L_6 \S L_{36}$ effectively becomes a single loop system L_1 , because there are not enough modes for the longer loops to have any effect. However, as m increases to 7, the second loop becomes active, and we see a jump in complexity. A similar situation repeats for the transition from $m = 36$ to 37, where the third loop becomes active.

The memory complexity starts to saturate once the total number of modes reaches the value R . Recall that R is the maximum number of modes in each component P_a of the progressive decomposition, in this case $R = 44$. In the region of m slightly larger than R , the components P_a are reaching their maximum possible size and the complexity reaches a plateau shortly after. We gain little additional complexity by increasing m further, though for the system with $\ell = 6$, already $m \approx R$ is in the regime where our progressive algorithm fails even on a supercomputer.

5.3 Comparison of power-law architectures

We generalize our analysis to three-loop power-law architectures from Definition 2.2 where the base length ℓ may be different and we study how this affects the complexity. Similarly to the previous section, we also vary the total number of modes m , and we use the input state $|\underline{n}\rangle = |1, 0, 1, 0, \dots\rangle$ with $\lceil \frac{m}{2} \rceil$ photons. We again compare the memory complexity to the capacity of a state-of-the-art supercomputer to better contextualize the potential for quantum advantage.

We present the results in Figure 12 where we show several colored series of data, each corresponding to a different value of ℓ . Similarly to the previous Fig. 11, we show the average and 95-th percentile memory as functions of m . We find that three-loop power-law systems with base length $\ell < 5$ and the above input are classically simulable using a large enough computer, even for a large number of modes. Thus we consider them not complex enough to reach quantum advantage.

⁴KDE uses an important parameter called *bandwidth*, and we use the *Scott's rule* [24] to automatically select it.

Memory complexity of $L_1 \circ L_6 \circ L_{36}$ (log scale) vs. number of modes

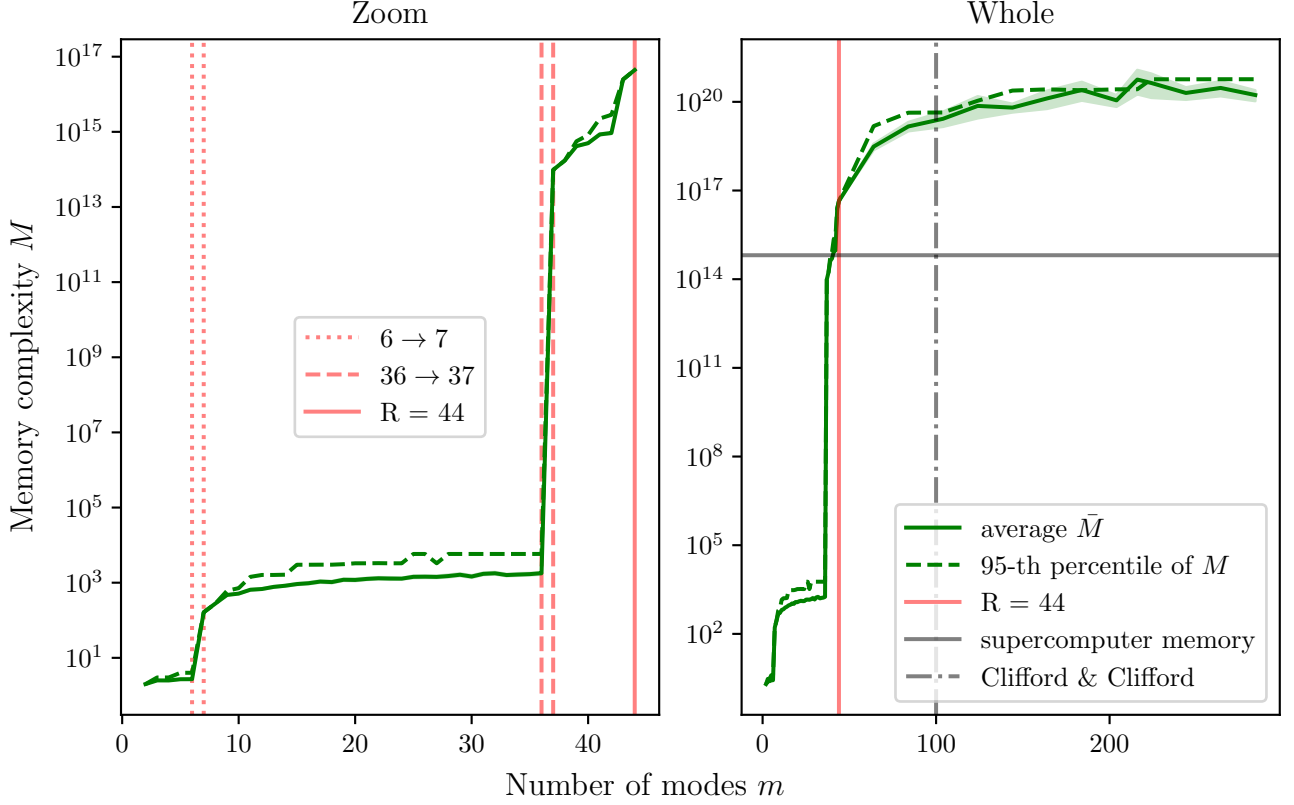


Figure 11: Memory complexity of simulating a loop-based system of lengths $\underline{\ell} = (1, 6, 36)$ with a varying number of modes m between 2 and 284 along the horizontal axis. We consider an input state of $|1, 0, 1, 0, \dots\rangle$ where there are half as many photons as modes. On the vertical axis, we show the memory complexity estimated using the heuristic p_H . We show two subplots of the same data, where the LHS picture only shows the range of m between 2 and 44 to make this region more visible. In each plot, we show the mean memory $\bar{M}$ as a full line and the 95-th percentile of the samples of memory as a dashed line. We observe that the memory increases abruptly between three regions, corresponding to each loop entering the simulation. We also observe that the complexity starts to saturate after $m = R = 44$, shown by the full red line. For comparison, we plot an estimate of the total memory available to a state of the art supercomputer (horizontal grey line), assuming that each coordinate is encoded as a double-precision complex number (16 bytes). We highlight the number of modes ($m = 100$ with our input) up to which the Clifford & Clifford algorithm can simulate the circuit (vertical dash-dotted line).

When $\ell = 5$, we reach a threshold in complexity. Observe in Figure 12 that the memory requirement to simulate a system with $\ell = 5$ with $\lceil \frac{m}{2} \rceil$ photons using our progressive simulator just surpasses the capacity of a state-of-the-art supercomputer. We see this happen when $m \geq 80$.

It follows that $\ell = 5$ is a lower bound for quantum advantage: Systems with $\ell < 5$ are clearly simulable by the progressive simulation algorithm. However, while systems with $\ell \geq 5$ are not simulable by our algorithm, they may still be simulable using other methods. For instance, in addition to having a sufficiently complex loop structure a quantum system must also have roughly 40 to 50 photons at least to avoid classical simulability with the algorithm of Clifford and Clifford [11].

We also highlight this threshold, which falls on $m = 100$ modes for our input $|1, 0, 1, 0, \dots\rangle$, in Figure 12. Moreover, as we discuss in the next section, experimental imperfections such as photon loss and distinguishability can reduce the requirements for classical simulation.

6 Discussion and conclusion

Here, we discuss ways in which our classical simulation method could be extended to broader settings beyond ideal boson sampling with single photons.

Memory M vs. modes and loop lengths in power-law systems (log scale)

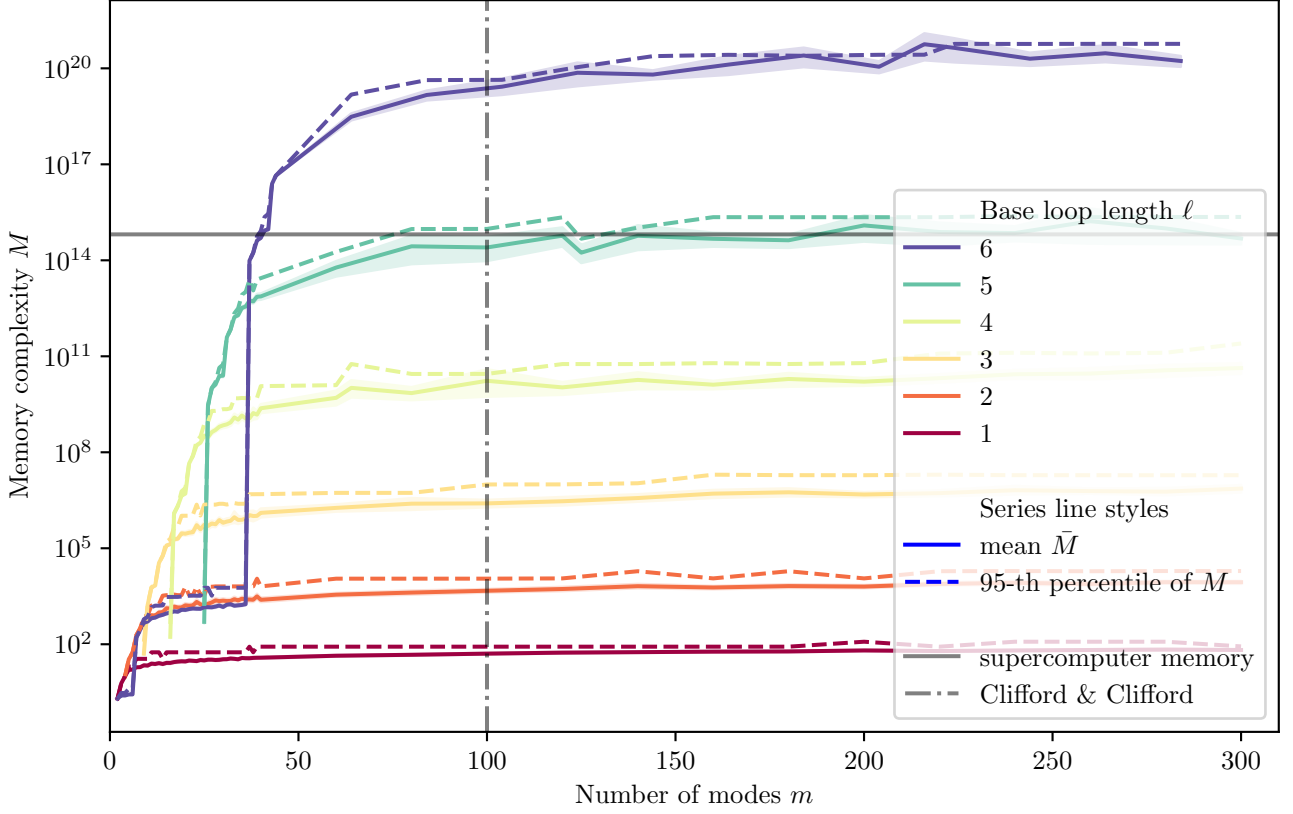


Figure 12: Comparison of power-law architectures $\underline{\ell} = (1, \ell, \ell^2)$ for different values of the base loop length ℓ , signified by the colors of the lines. We consider an input state of $|1, 0, 1, 0, \dots\rangle$ where there are half as many photons as modes. The plot is similar to Figure 11: we show the total number of modes on the horizontal axis, the memory complexity on the vertical axis, and we plot the mean $\bar{M}$ and 95-th percentile, comparing them to the supercomputer memory drawn as horizontal grey line. We observe that $\ell = 5$ is a threshold for complexity when using the progressive simulator. We furthermore highlight the number of modes ($m = 100$ with our input of $n = 50$ photons) up to which the Clifford & Clifford algorithm succeeds in simulating the circuit. Thus we divide the plot into quadrants representing regimes where different simulators are appropriate, and we observe the bottom right quadrant containing systems simulable by our algorithm, but not by the Clifford & Clifford algorithm.

Extension to loss Here, we show that this method can be extended to simulate lossy circuits. Loss on a mode can be modeled as a beam splitter between the mode and the environment. We can thus evolve an initially n -mode pure state through loss on mode a by applying a beam splitter between mode a and the environment. We then simulate a measurement process on the environmental mode, by calculating the probabilities of measuring different photon numbers on the environmental mode and sampling from these possible outcomes. This measurement collapses the initial n modes back to a pure state with potentially fewer photons. Though the probabilistic outcome of the environmental measurement makes this process stochastic, the resulting statistics match those obtained by tracing out the

environmental mode and using the density matrix formalism. However, our method avoids the quadratic overhead stemming from the use of a density matrix. Repeating this process at every loss location in the circuit allows us to exactly simulate sampling from the lossy state using our classical simulation algorithm.

Moreover, the complexity of the lossless case provides an upper bound to the complexity of simulating a lossy circuit. Adding additional beam splitters to simulate loss adds an overhead which is only linear in the number of optical components. However, the state space is also significantly smaller than the lossless case since we now simulate a state with fewer photons. Though our lattice path formalism for determining the exact size of the state space cannot be directly applied

in the case of loss, the lossless case thus provides an upper bound.

Extension to distinguishable photons Our method can also be extended to simulating boson sampling with partial distinguishability. Following [26, 27], a simple model for a system with partially distinguishable photons involves considering that each photon is in a superposition of being in an indistinguishable mode and a distinguishable mode. Sampling from this model can be simulated by randomly assigning the input photons into a distinguishable set and an indistinguishable set at the beginning of the simulation, with an assignment probability determined by the initial superposition state. Evolving the indistinguishable photons through the circuit and collecting the output measurements, then simulating the distinguishable photons and summing both measurements, would yield the correct output statistics. The complexity of this method is dominated by the complexity of simulating this subset of indistinguishable photons, which is upper-bounded by the complexity of simulating the full set of indistinguishable photons given by our lattice path formalism. We note that more involved models for partial distinguishability have been developed, which we anticipate could also be used within our simulation method [28, 29].

Gaussian Boson Sampling We focus on SPBS, but there are other variations of Boson Sampling that have been investigated, such as Gaussian Boson Sampling (GBS). For example, in Section 5.2, we studied the system of loops $\ell = (1, 6, 36)$ which was inspired by a GBS advantage experiment by Madsen *et al.* [6]. GBS is underpinned by a different formalism than SPBS, so our methods are not directly transferable. However, it would be interesting to explore whether the staircase structure of the interferometer (Figure 1a) could be exploited to accelerate GBS simulations.

6.1 Summary and outlook

In this work, we study the potential for quantum advantage in loop-based boson sampling systems, and to this end, we provide new tools to quantify their classical simulation complexity. First, we exploit the circuit topology to design a new simulation method that requires that only small

subcircuits are strongly simulated. This can use any wavefunction Fock basis simulator, which implies that a relevant choice of complexity metric is the memory required to store the state, i.e. the dimension of the reachable state space.

To fully and efficiently characterize the reachable state space, we extend the lattice path formalism of Brádler & Wallner [10]. Our extension can encode the action of beamsplitters between arbitrary modes, and similarly the outcomes of measurements of arbitrary modes. To use the formalism to predict the memory complexity of simulating large systems, we define a heuristic probability on the possible outcomes that is easy to sample from. We validate the statistics given by the heuristic by comparing its predictions to empirical data from simulations of small to medium-scale systems.

Using these tools, we study the complexity of power-law architectures where the loops have lengths $1, \ell, \ell^2$ and we use input state $|1, 0, 1, 0, \dots\rangle$, to represent the common experimental scenario where there are half as many photons as modes. A system with $\ell < 5$ (and any number of modes m) can be efficiently simulated using our method, and thus does not achieve advantage. We provide a lower bound for advantage in such systems at $\ell \geq 5$.

To conclude, our work provides new insight into the boundaries for quantum advantage using single photons in loop-based time-bin interferometers. We develop new tools that provide new lower bounds for the arrangement of the loops. We anticipate that this work will help guide future experimental implementations of boson sampling, and points the way for further research into quantum advantage with loop-based systems under different experimental scenarios.

Acknowledgements

We would like to thank the members of the team at ORCA for helpful discussions. This work was supported by the Quantum Data Centre of the Future, project number 10004793, funded by the Innovate UK Industrial Strategy Challenge Fund (ISCF). R.G.-P. was supported by the EPSRC-funded project Benchmarking Quantum Advantage.

A Results on the loop-based structure

To formally prove Theorem 3.1, we formalize the notion of travelling through a circuit X backwards. For an output mode a , we denote $X^\dagger(a)$ the set of all input modes, photons from which can reach the output a with nonzero probability, constrained only by the circuit topology. That is, the set $X^\dagger(a)$ is a preimage of the relation encoding connectivity from input to output modes.

We recall the statement of the theorem below:

Theorem 3.1 (relevant modes). In a progressive decomposition of a loop-based system X made of loops of lengths $\underline{\ell} = (\ell_1, \dots, \ell_\Lambda)$, each component P_a acts non-trivially on at most the number of *relevant* modes

$$R = 1 + \sum_{i=1}^{\Lambda} \ell_i. \quad (3.1)$$

Proof. Recall that a component P_a is made of beamsplitters relevant to measuring output mode a . We use these beamsplitters to reach the modes in $X^\dagger(a)$ while traversing the circuit backward: a beamsplitter allows us to jump to a different mode, as shown in the example of Figure 4.

To bound the number of relevant modes, we need to identify the furthest reachable mode, i.e. the one identified by the maximum number $\max X^\dagger(a)$. The mode $\max X^\dagger(a)$ is found by starting at output a and following backward the circuit $X = L_1 \circ L_{\ell_2} \circ \dots \circ L_{\ell_\Lambda}$, taking a step $+\ell_j$ (towards a mode identified by a higher number) at each loop L_{ℓ_j} ; for example, see the red path with arrows $\Leftarrow$ in Figure 4. This way, reach the maximum mode: $\max X^\dagger(a) = a + \sum_{j=1}^{\Lambda} \ell_j$.

Finally, recall that by construction of the progressive decomposition, the component P_a does not act on modes $\llbracket 0, a-1 \rrbracket$: Any beamsplitters affecting these are captured by $P_0, \dots, P_{a-1}$, and furthermore, the modes $\llbracket 0, a-1 \rrbracket$ are measured before the component P_a is simulated. Having previously identified the maximum reachable mode $\max X^\dagger(a)$, we conclude that the maximum number of modes that P_a can act on is:

$$\left| \llbracket a, a + \sum_{j=1}^{\Lambda} \ell_j \rrbracket \right| = 1 + \sum_{j=1}^{\Lambda} \ell_j. \quad \blacksquare$$

Finally, we note that this upper bound is saturated whenever a component P_a starts with

a staircase of nearest-neighbour beamsplitters given by L_1 – namely, this is the case for P_0 in a system with $\ell_1 = 1$. The reason is that the set of inputs reachable via L_1 from any output mode b is $L_1^\dagger(b) = \llbracket 0, b+1 \rrbracket$. This is because the staircase of nearest neighbour beamsplitters allows us to take one step from b to $b+1$, and as many steps towards lower ($< b$) modes as we like, as shown in Figure 4 by the yellow region.

Other progressive components (P_a for $a > 0$) in a system with $\ell_1 = 1$, and more generally any component if $\ell_1 \neq 1$, may act on fewer modes than R . The procedure of travelling through the circuit backward, and using beamsplitters to take steps up or down, yields a set which is not an interval, but has gaps. We do not need more detail on this for our work, however, the set $X^\dagger(a)$ can be exactly characterized using the rich theory of numerical monoids, see [30].

B Formal details of the lattice path formalism

In order to give a more formal proof of Theorem 4.4, particularly the part where we merge contracted skew diagrams, we need to fill in some details omitted in Sections 2.3.3 and 4. We need tools from order theory: we present generalities inlined in the text, and we only number definitions or lemmas specific to our work.

B.1 Young order

We start by recalling that a *partial order* $\leq$ on a set P is a generalization of a total order (e.g. the usual ordering on the real line $\mathbb{R}$) where some elements may be incomparable. The set P is a *partially ordered set* (*poset*); for more details, see [31]. The order relevant to us is the following:

Definition B.1 (Young order). Take two paths $\lambda, \kappa \in \mathcal{L}(W, H)$ in height representation. We define a partial order, called the *Young order*, where $\lambda \leq \kappa$ iff $\lambda_a \leq \kappa_a$ for each mode a .

It is called the Young order because lattice paths can be seen as generalized Young diagrams when we fill the boxes below them.⁵ Then the order can be understood geometrically as the in-

⁵Note the choice of orientation is such that lattice paths are nondecreasing sequences.

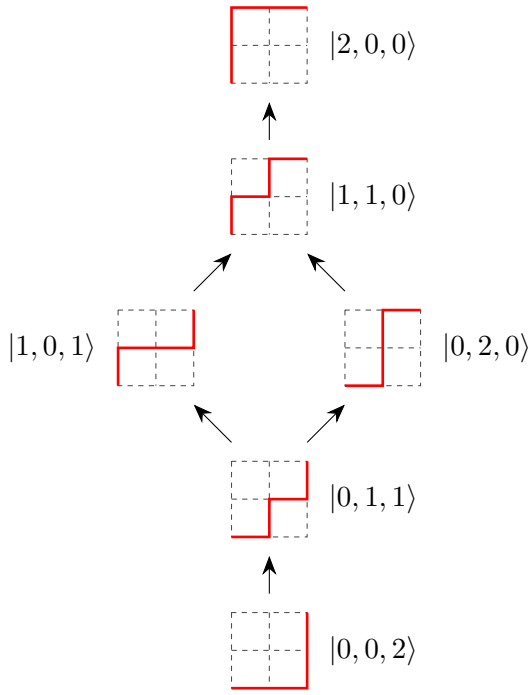
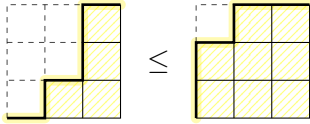


Figure 13: The Hasse diagram of the set of paths $\mathcal{L}(2, 2)$, which corresponds to the bosonic Fock space $\mathcal{F}^2\mathbb{C}^3$, with the Young order. The arrows show the covering relation $\lambda \leq \kappa$. Next to each lattice path, we display the number state it represents.

clusion of Young diagrams; we show an example inequality below:



The Young order is well-known in the literature and has many properties useful to us later [31].

We present an example of the order for $\mathcal{L}(2, 2)$ in Figure 13 which shows the order's Hasse diagram: a directed graph where vertices are elements of $\mathcal{L}(2, 2)$ and there is an arrow $\lambda \rightarrow \kappa$ iff λ is covered by κ (written $\lambda \leq \kappa$). The latter means that $\lambda < \kappa$ and there is no element between them. In the geometric picture, $\lambda \leq \kappa$ when the Young diagram of λ has exactly one box less than κ .

B.2 Representing measurements

The part of the formalism that encodes measurements uses *intervals* in the Young poset, which in our paper are called *skew diagrams* in analogy to skew Young diagrams. Recall that an interval is

a subset of the poset defined by two paths $\nu \leq \mu$:

$$\mu/\nu \equiv [\nu, \mu] := \{\lambda : \nu \leq \lambda \leq \mu\}.$$

When necessary, we write it vertically as $\frac{\mu}{\nu}$.

B.2.1 Projection to skew diagrams

Recall that to project to a subspace compatible with some measurement, say $n_a = x$, in Section 4.2.1, we impose an S-shaped region onto the original downset $\mu\downarrow$. For the following steps, it is useful to always reason in terms of skew diagrams; note that the downset is the skew diagram $\mu/\perp$ where $\perp$ is the bottom element of the poset, i.e. the path $\lambda^\perp = (0, \dots, 0, n)$.

The S-shape is parametrized by its starting height, controlling the number of photons in the modes to the left of the S, thus implementing the constraint $\lambda_{a-1} = q$ on the lattice paths λ in the post-measurement subspace. It then takes a step of size x , encoding the number of photons found in mode a , implementing the requirement that $n_a = x$. Away from the S-shape, the skew diagram should stay as unchanged as possible from the original downset. A convenient way to construct this is to define *measurement helpers*, skew diagrams which only encode information about the measurement, and leave everything else as free as possible. We show an example in Figure 14, and we precisely define helper diagrams below. Then we take an intersection of the helper with the original skew diagram $\mu/\perp$ to obtain a projection.

Definition B.2 (measurement helper). To represent a subspace of the full Fock space $\langle \mathcal{L}(m-1, n) \rangle_{\mathbb{C}}$ that has in total $\lambda_{a-1} = q$ photons in modes $[0, a-1]$ and x photons in mode a , define a *helper skew diagram* $E(q) = \varepsilon^q/\eta^q$ where the top path is

$$\varepsilon^q = (\underbrace{q, \dots, q}_{a-1}, \underbrace{q+x, \dots, q+x}_{m-a-1}, n)$$

and the bottom is

$$\eta^q = \begin{cases} (\underbrace{0, \dots, 0}_{a-1}, \underbrace{q+x, \dots, q+x}_{m-a-1}, n) & \text{if } a > 0, \\ (\underbrace{q+x, \dots, q+x}_{m-a-1}, n) & \text{if } a = 0. \end{cases}$$

Note that if $a = 0$, then only $E(q = 0)$ exists, so the above second case becomes $\eta^0 = (x, \dots, x, n)$. Note also that the maximum q is $q_{\max} = n - x$.

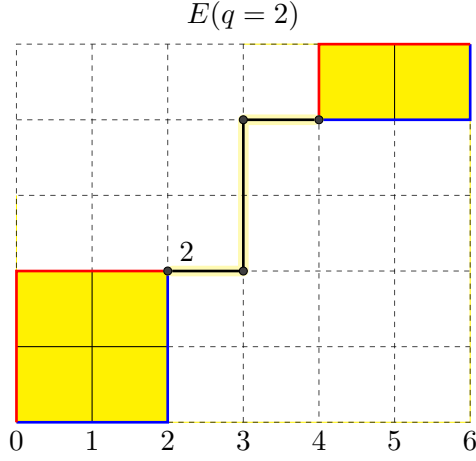


Figure 14: Helper diagram $E(q=2)$ which represents the subspace of the full Fock space $\langle \mathcal{L}(m-1, n) \rangle_{\mathbb{C}}$ (with $m=7$ and $n=5$) of those basis states that have $q=2$ photons in modes $\llbracket 0, 2 \rrbracket$, represented by the starting position of the S-shaped region, and $x=2$ photons in mode 3, represented by the height of the step in the S-shape. No other constraints are implemented in this skew diagram.

The notation $E(q)$ does not specify the mode a , nor the number of photons x , however, in the following text, this will always be clear from the context.

The above definition is the first step to formally define the projection step in terms of lattice paths, i.e. in a way that can be written as a formula (and implemented in code). We use it to represent the subset of the basis of a full Fock space that remains after a given measurement. The next step is to take its *intersection* with the diagram $\mu/\perp$ representing the state space of our system before measurement to obtain those basis states that agree with both. In order to obtain the intersection of skew diagrams, we first need some more theory of the Young order:

The poset $\mathcal{L}(m-1, n)$ is an order-theoretic *lattice* which means that it contains all finite meets and joins [31]. Recall that a *meet* (also called *supremum* or *least upper bound*) of λ and κ is the unique least element $\lambda \wedge \kappa$ such that $\lambda \leq \lambda \wedge \kappa$ and $\kappa \leq \lambda \wedge \kappa$. Dually, the *join* (also *infimum* or *greatest lower bound*) is the unique greatest element $\lambda \vee \kappa$ such that both λ and κ are greater.

Lemma B.3 (meet and join in Young order). Suppose $\lambda, \kappa \in \mathcal{L}(m-1, n)$. Then their meet and

join have the following components:

$$\begin{aligned} (\lambda \wedge \kappa)_a &= \min\{\lambda_a, \kappa_a\}, \\ (\lambda \vee \kappa)_a &= \max\{\lambda_a, \kappa_a\}. \end{aligned}$$

Proof. Recall that in the Young order $\alpha \leq \beta$ iff $\alpha_a \leq \beta_a$ for all a , making it a product order of several copies of the linear order on integers. This means the meet and join are componentwise the meet and join of integers, which are the minimum and maximum, respectively. Note that this is compatible with the fact that the height representation of a lattice path must be a nondecreasing sequence: a component a is a meet (resp. join), and so is $a+1$, etc., so the nondecreasing constraint is satisfied. ■

Lemma B.4 (intersection of skew diagrams). Suppose α/β and γ/δ are skew diagrams in $\mathcal{L}(m-1, n)$. Their intersection is either empty or equal to the following:

$$\frac{\alpha}{\beta} \cap \frac{\gamma}{\delta} = \frac{\alpha \wedge \gamma}{\beta \vee \delta}.$$

Proof. The intersection must contain exactly the paths λ such that $\beta \leq \lambda \leq \alpha$ and simultaneously $\delta \leq \lambda \leq \gamma$. It is possible that no such λ exists, in which case the intersection is empty. If the intersection is nonempty, then it is clearly an interval (skew diagram). The greatest λ that satisfies both $\lambda \leq \alpha$ and $\lambda \leq \gamma$ is the meet $\alpha \wedge \gamma$. Dually, the least λ is the join $\beta \vee \delta$. Note that this relies on $\mathcal{L}(m-1, n)$ being an order lattice. ■

With the above tools, we can characterize the state spaces after the measurement projection:

Lemma B.5. The family of skew diagrams obtained from μ/ν after the projection implementing the measurement of x photons in mode a consists of skew diagrams parametrized by the number of photons q in modes $\llbracket 0, a-1 \rrbracket$ written as follows:

$$\frac{\mu^q}{\nu^q} = \frac{\mu}{\nu} \cap E(q) = \frac{\mu \wedge \varepsilon^q}{\nu \vee \eta^q}.$$

Proof. The intersection selects those lattice paths λ in the original state space μ/ν which also satisfy $\lambda_{a-1} = q$ and $n_a^\lambda = x$, the latter constraints imposed by the measurement helper $E(q)$. The second equality follows from Lemma B.4. ■

Note that the values of q which yield nonempty intersections μ^q/ν^q come from an interval $\llbracket q_{\min}, q_{\max} \rrbracket$ whose endpoints depends on μ and ν and x . Notably, in our case of $\nu = \perp$, these are $q_{\min} = 0$ and $q_{\max} = \min\{\mu_{a-1}, \mu_a - x\}$.

B.2.2 Contraction

Having the projection, we now formalize the next step, the contraction of the projected skew diagrams from Section 4.2.2. The contraction step is required to make the merge step in the next section work. Recall that the contraction removes a mode and all the photons contained therein.

Definition B.6 (contraction). In a poset $\mathcal{L}(m-1, n)$, measuring x photons in mode a and then deleting it is represented by the function

$$C_a^x : \bigcup_q E(q) \rightarrow \mathcal{L}(m-2, n-x)$$

that sends

$$\begin{array}{c} (n_0, \dots, n_{a-1}, x, n_{a+1}, \dots, n_{m-1}) \\ \downarrow \\ (n_0, \dots, n_{a-1}, \quad n_{a+1}, \dots, n_{m-1}) \end{array}$$

deleting the component a in step representation and preserving the others. The q in $\bigcup_q E(q)$ runs over all allowed values $\llbracket 0, n-x \rrbracket$. The domain is the subset of $\mathcal{L}(m-1, n)$ where the contraction makes sense.

Recalling Eq. (2.3), the height representation is transformed as follows:

$$(C_a^x \lambda)_b = \begin{cases} \lambda_b & \text{if } b \leq a-1, \\ \lambda_{b+1} - x & \text{if } b \geq a. \end{cases} \quad (\text{B.1})$$

What follows are two technical lemmas showing that the contraction is well-behaved with respect to the Young order: it is an order isomorphism, which means it is a bijection of sets and preserves the order structure of those sets. This is important because it is this order structure that allows us to efficiently count basis states.

Lemma B.7 (contraction is monotone). The contraction function $C_a^x : \bigcup_q E(q) \rightarrow \mathcal{L}(m-2, n-x)$ preserves the Young order.

Proof. Take two paths $\lambda, \kappa \in \bigcup_q E(q)$, and suppose $\lambda \leq \kappa$. Equivalently $\lambda_b \leq \kappa_b$ for each b . The contraction deletes mode a and all the photons contained therein. Following Equation (B.1), for modes $b \leq a-1$, we have

$$(C_a^x \lambda)_b = \lambda_b \leq \kappa_b = (C_a^x \kappa)_b.$$

Similarly, for modes $b \geq a$, we have

$$(C_a^x \lambda)_b = \lambda_{b-1} - x \leq \kappa_{b-1} - x = (C_a^x \kappa)_b.$$

We conclude that $C_a^x \lambda \leq C_a^x \kappa$. ■

Lemma B.8 (contraction is isomorphism). The contraction function C_a^x is an order isomorphism, i.e. a monotone bijection $\bigcup_q E(q) \simeq \mathcal{L}(m-2, n-x)$ whose inverse is also monotone.

Proof. The function C_a^x is clearly injective: it forgets the a -th component of a lattice path in the step representation, but that component is, by construction, always $n_a = x$ for the paths contained in the domain $\bigcup_q E(q)$. Thus it forgets no information, and is clearly invertible, at least from its image, written $\bigcup_q \tilde{E}(q)$, where each $\tilde{E}(q) = C_a^x(E(q))$. Writing its inverse $(C_a^x)^{-1} : \bigcup_q \tilde{E}(q) \rightarrow \bigcup_q E(q)$ is easy in the step representation: simply insert a component of value x at position a to undo the action of C_a^x . We know from Lemma B.7 that C_a^x preserves order, and by a similar argument, we see that $(C_a^x)^{-1}$ does as well.

What remains to be shown is that C_a^x is surjective, i.e. that $\bigcup_q \tilde{E}(q) = \mathcal{L}(m-2, n-x)$. Take an arbitrary path $\lambda \in \mathcal{L}(m-2, n-x)$. We show that we can act by the inverse $(C_a^x)^{-1}$. Inserting a component x at position a into $\underline{n}^\lambda$ is clearly possible and sends λ to $E(q = \lambda_{a-1})$. Finally, we note that $E(q = \lambda_{a-1})$ is indeed a part of the domain $\bigcup_q E(q)$: recall that q ranges over $\llbracket 0, n-x \rrbracket$, as does λ_{a-1} . We conclude that C_a^x is a bijection of sets, and an order isomorphism. ■

B.2.3 Merge

The final step is to merge the contracted skew diagrams into a single skew diagram, and indeed a downset, containing exactly the correct paths. We fill in the details omitted in Section 4.2.3. Skew diagrams are intervals in the poset, and merging them means taking their union as sets:

$$\bigcup_q C_a^x \left(\frac{\mu^q}{\nu^q} \right).$$

It is not entirely trivial to see that such a union is also an interval. Indeed, this is not the case when there is a gap between the intervals, i.e. there exist elements between them. To see the intuition, we use an example of the real line with its usual order: Suppose we have two intervals $[a, b]$ and $[c, d]$ s.t. $b \leq c$. If $b < c$, then there exists some $x \in (b, c)$ that is not contained in either interval, nor in their union $[a, b] \cup [c, d]$. This means that the union is not itself an interval.

One can easily find examples to show that, without contraction, the union $\bigcup_q E(q)$ (and therefore $\bigcup_q \frac{\mu^q}{\nu^q}$) is not generally an interval. However, the contraction C_a^x removes the gaps between consecutive helper diagrams, i.e. intervals $\tilde{E}(q)$ and $\tilde{E}(q+1)$, where we recall the notation $\tilde{E}(q) \equiv C_a^x(E(q))$. The strategy to show this is to study the boundary between a contracted helper $\tilde{E}(q+1)$ and a hypothetical gap between $\tilde{E}(q)$ and $\tilde{E}(q+1)$, showing that such gap cannot exist.

Boundaries of intervals To do this, define a *lower boundary* of a subset S of a poset P as

$$\partial S = \{(\lambda, \kappa) : \kappa \in S, \lambda \in P \setminus S, \lambda \leq \kappa\}.$$

If S is a contiguous region in the Hasse diagram, then ∂S can be seen as a cut between S and the region outside (and lower than) S . We can dually define an *upper boundary*, but the reasoning will be analogous, so it is enough to focus on ∂S .

The strategy is to show that for a contracted interval, the lower point λ of the boundary belongs to another contracted interval. We start with a general statement:

Lemma B.9. Let α/β be some skew diagram, and let $(\lambda, \kappa) \in \partial(\alpha/\beta)$, then there exists a component b such that

$$\lambda_c = \begin{cases} \beta_b - 1 & \text{if } c = b, \\ \kappa_c & \text{if } c \neq b. \end{cases}$$

Proof. Recall that if $\lambda \leq \kappa$, then geometrically λ has one fewer box than κ as a Young diagram. In terms of the height representation, there must be exactly one component, here b , where they differ: $\lambda_b = \kappa_b - 1$. By definition of the boundary, $\kappa \in \alpha/\beta$, thus $\kappa_c \in \llbracket \beta_c, \alpha_c \rrbracket$ for each c . On the other hand, $\lambda \notin \alpha/\beta$, so there must exist at least one c for which $\lambda_c \notin \llbracket \beta_c, \alpha_c \rrbracket$, which implies $\lambda_c < \beta_c$ since $\lambda \leq \kappa$. Since λ and κ must agree on all but one element, the component c where this happens is $c = b$. Together, this implies that $\kappa_b = \beta_b$ and $\lambda_b = \beta_b - 1$. ■

Lemma B.10. Let $(\lambda, \kappa) \in \partial C_a^x(E(q+1))$. Then the value b from Lemma B.9 is $b = a - 1$.

Proof. First, we show that $b \geq a - 1$ which follows from Lemma B.9: we have $\lambda_b = C_a^x(\eta^{q+1})_b - 1$, and by Definition B.2, for $c \in \llbracket 0, a - 2 \rrbracket$, the bottom path has components $\eta_c^{q+1} = C_a^x(\eta^{q+1})_c =$

0. The value λ_b must be nonnegative, and thus $b \geq a - 1$.

Next, we show that $b \leq a - 1$. By case-by-case analysis, we show that (λ, κ) in the boundary of a contracted helper diagram can not exist otherwise. We display the cases in the table below, omitting their individual derivation which is simple but tedious. Each case shows the allowed values of λ_c depending on the choice of b and $c \neq b$.

	$c < b$	$c > b$
$c \in \llbracket 0, a - 2 \rrbracket$	$\llbracket 0, q \rrbracket$	$\{q, q + 1\}$
$c = a - 1$	$\emptyset$	$\{q + 1\}$
$c \in \llbracket a, m - 3 \rrbracket$	$\emptyset$	$\llbracket q + 1, n - x \rrbracket$
$c = m - 2$	$\emptyset$	$\{n - x\}$

Cells which violate order requirements are shown by $\emptyset$ to signify that they represent cases with no possible value for λ_c . Conclude that the only way to have a value of λ_c for all c is if $b \leq a - 1$. ■

Lemma B.11. Let $(\lambda, \kappa) \in \partial C_a^x(E(q+1))$. Then $\lambda \in C_a^x(E(q))$.

Proof. This follows from Definition B.2 and Lemmas B.9 and B.10. We know that

$$\lambda_{a-1} = \tilde{\eta}_{a-1}^{q+1} - 1 = q = \tilde{\eta}_{a-1}^q,$$

where we denote $\tilde{\alpha} \equiv C_a^x(\alpha)$ for paths α . For all $c \neq a - 1$, we have $\lambda_c = \kappa_c \in \llbracket \tilde{\eta}_c^{q+1}, \tilde{\varepsilon}_c^{q+1} \rrbracket$. For $c < a - 1$, this is $\llbracket 0, q \rrbracket$, and for $c > a - 1$, it is $\llbracket q+1, n-x \rrbracket$. We conclude that $\lambda \in C_a^x(E(q))$. ■

Lemma B.12 (union of consecutive intervals). The union of consecutive contracted intervals $\tilde{E}(q)$ and $\tilde{E}(q+1)$ is

$$\tilde{E}(q) \cup \tilde{E}(q+1) = \frac{\tilde{\varepsilon}^{q+1}}{\tilde{\eta}^q}.$$

Proof. This requires three facts, two of which are immediate: First, the set $\tilde{E}(q) \cup \tilde{E}(q+1)$ has a maximum element $\tilde{\varepsilon}^{q+1}$, i.e. the top element of the interval $\tilde{E}(q+1)$. Similarly, it has a minimum $\tilde{\eta}^q$, i.e. the bottom element of $\tilde{E}(q)$. This follows from the construction of (contracted) helper diagrams.

The final step is to show that there are no elements between the intervals $\tilde{E}(q)$ and $\tilde{E}(q+1)$. If λ is between them, there exist $\alpha \in \tilde{E}(q)$ and $\beta \in \tilde{E}(q+1)$ s.t. $\alpha < \lambda < \beta$. It is possible to find such λ that is adjacent to $\tilde{E}(q+1)$ in the Hasse diagram, i.e. it is covered by a $\kappa \in \tilde{E}(q+1)$. This is equivalent to saying $(\lambda, \kappa) \in \partial \tilde{E}(q+1)$.

From Lemma B.11, it follows that $\lambda \in \tilde{E}(q)$. We conclude that there is no gap between the two consecutive contracted helper intervals, and thus their union is the interval $\tilde{\varepsilon}^{q+1}/\tilde{\eta}^q$. ■

B.2.4 Full description of measurement

The result of Lemma B.12 clearly extends to the whole union

$$\bigcup_q \tilde{E}(q) = \frac{\tilde{\varepsilon}^{q_{\max}}}{\tilde{\eta}^{q_{\min}}} = \frac{\tilde{\varepsilon}^{n-x}}{\tilde{\eta}^0}.$$

Furthermore, it extends to the union of contracted intersections $\bigcup_q C_a^x(\mu/\nu \cap E(q))$: each intersection is taken with the same skew diagram μ/ν , and the effect is simply a further restriction on modes away from $a - 1$.

Finally, having proved all that is needed for Theorem 4.4 in the preceding sections, we recall its statement below:

Theorem 4.4 (measurement). Starting with a cumulative space $\langle \mu \downarrow \rangle$, the state space left after measuring x photons in mode a is a cumulative space $\langle \mu^{(n_a=x)} \downarrow \rangle$ where the maximal path $\mu^{(n_a=x)}$ is the contraction of $\mu^{q_{\max}}$. The latter is the maximum path of the skew diagram corresponding to $q = q_{\max} = \min\{\mu_{a-1}, \mu_a - x\}$.

B.3 Coupling new modes in initial L_1

In Section 4.3, we mention a subtlety in applying beamsplitters from the initial loop L_1 . The issue is that the initial loop couples a PCS to a new, previously unseen mode, with new photons; thus the rule from Theorem 4.2 does not apply.

Though it is likely not possible in general to couple an additional mode to an arbitrary mode of a PCS, it is possible in the special case of coupling the maximum mode of a PCS to a new mode via a beamsplitter of the initial L_1 . Suppose we start with m modes and n photons. All we need to do is expand the lattice rectangle width by one mode, height by the number x of new photons in the added mode, and to set the height of μ in the final two modes to the new total number of photons $n + x$:

$$(\mu_0, \dots, \mu_{m-2}, n) \mapsto (\mu_0, \dots, \mu_{m-2}, n + x, n + x).$$

B.4 Counting lattice paths

We give a dynamic programming algorithm to efficiently count a downset $\mu \downarrow$ or a skew diagram

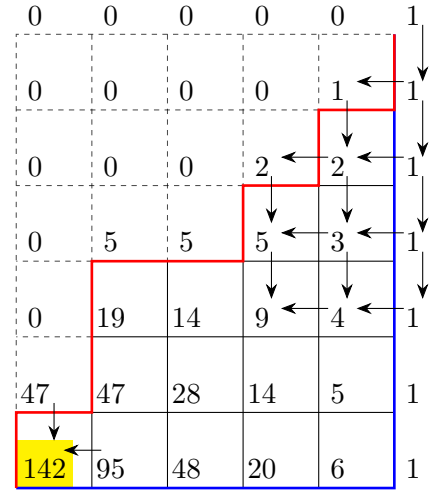


Figure 15: A schematic explanation of the counting algorithm for an example skew diagram μ/ν (in fact, a downset, i.e. with ν the bottom path). To each point (x, y) in the lattice rectangle, we associate a number $A_{x,y}$. This is the number of allowed lattice paths within a part of the skew diagram that start at point (x, y) and terminate, as usual, in the top right corner (W, H) . The array A is what Algorithm 1 computes. The total number of paths in the diagram is found at point $(0, 0)$, i.e. $|\mu/\nu| = A_{0,0}$.

Algorithm 1: count_interval

Input: Lattice paths $\nu \leq \mu$ identifying a skew diagram μ/ν .

Output: Cardinality $|\mu/\nu|$.

```

1  $m \leftarrow$  length of  $\mu$ ;  $n \leftarrow \mu_{m-1}$ 
2 if  $m = 1$  or  $n = 0$  then return 0
3 Initialize array  $A = (A_{i,j})_{i,j}$  with all zeros,
   for  $i = 0, \dots, m - 1$  and  $j = 0, \dots, n$ 
4  $A_{m-1,n} \leftarrow 1$ 
5 for  $x \leftarrow m - 1, m - 2, \dots, 0$  do
6   if  $x = 0$  then  $y_{\min} \leftarrow 0$ 
7   else  $y_{\min} \leftarrow \nu_{x-1}$ 
8   for  $y \leftarrow \mu_x, \mu_x - 1, \dots, y_{\min}$  do
9      $A_{x,y} \leftarrow A_{x+1,y} + A_{x,y+1}$  where
        $A_{i,j} = 0$  if  $i$  or  $j$  are out of bounds
10 return  $A_{0,0}$ 

```

Figure 16: Pseudocode of the dynamic programming algorithm to count the lattice paths in a given skew diagram. For notational simplicity, we allow the indices i, j of $A_{i,j}$ to be out of bounds ($i > m - 1$ or $j > n$) and we assume $A_{i,j} = 0$ in those cases.

μ/ν . To formulate it, we recall that a lattice path

can be understood as a sequence of steps right $(+1, 0)$ or up $(0, +1)$.

Note that given a point (x, y) in the lattice rectangle $\llbracket 0, W \rrbracket \times \llbracket 0, H \rrbracket$, the number of paths passing through this point is

$$|\mathcal{L}(x, y)| \cdot |\mathcal{L}(W - x, H - y)|.$$

This is the number of paths that reach (x, y) , times the number of paths that start at (x, y) and reach (W, H) . To count the paths in μ/ν , we restrict the above sets to regions allowed by the skew diagram.

Together, the two ideas above give us an efficient algorithm to compute the number of lattice paths in a skew diagram μ/ν ; see example in Figure 15 and pseudocode Algorithm 1 in Figure 16.

To each point (x, y) of the lattice rectangle $\llbracket 0, W \rrbracket \times \llbracket 0, H \rrbracket$, we associate an integer $A_{x,y}$ that is the number of paths starting at that point, ending at (W, H) , and staying within the region μ/ν . These form an array A , shown in Figure 15. From (x, y) , a path can go up to $(x, y + 1)$ or right to $(x + 1, y)$, and at each of those points, we have a corresponding value of A which counts the (disjoint) set of paths starting there. This allows us to count paths starting at (x, y) :

$$A_{x,y} = A_{x+1,y} + A_{x,y+1}.$$

Above, by abuse of notation, we assume that $A_{x,y} = 0$ if $(x, y) \notin \llbracket 0, W \rrbracket \times \llbracket 0, H \rrbracket$. Points outside of μ/ν also get $A_{x,y} = 0$. Finally, we conventionally define $A_{W,H} = 1$: there is a single (empty) path from (W, H) to itself.

To evaluate all entries of A , we start with $A_{W,H} = 1$ and progress through the array towards $(0, 0)$, evaluating an entry once its dependencies are available, as shown in Figure 15 and Algorithm 1. The final value is $A_{0,0} = |\mu/\nu|$.

$$[B_{ab}(\theta)]_{n_a, n_b}^{N_a, N_b} = \sqrt{n_a! n_b! N_a! N_b!} \sum_{k=\max\{0, N_b - n_a\}}^{\min\{n_a, N_a\}} \frac{(-1)^{n_a - k + 1} (\cos \theta)^{n_b + k - l} (\sin \theta)^{n_a - k + l}}{k! (n_a - k)! l! (n_b - l)!}, \quad (\text{C.2})$$

where k represents the number of photons starting in mode a and ending in the same mode a , and

C Theoretical complexity of the progressive sampling algorithm

In Section 1.1, we compared the asymptotics of our Algorithm PS to other algorithms in the literature. Here, we provide the derivation. This will concern our reference implementation of the algorithm, found in the companion Git repository.²

Since the algorithm is dynamic in nature, in that the computations for component P_{a+1} depends on the number of photons measured after component P_a , it would be difficult to estimate its theoretical time complexity in the general case. Thus here we focus on one kind of circuit of interest, the power-law architecture with loop lengths $\underline{\ell} = (1, \ell, \ell^2, \dots, \ell^{\Lambda-1})$. Note that in this case, the number of modes per components of the progressive decomposition, given by (3.1), is:

$$R = 1 + \sum_{i=1}^{\Lambda} \ell^{i-1} = \begin{cases} 1 + \Lambda & \text{if } \ell = 1, \\ 1 + \frac{\ell^{\Lambda} - 1}{\ell - 1} & \text{if } \ell > 1, \end{cases} \quad (\text{C.1a})$$

$$= \begin{cases} \Theta(\Lambda) & \text{if } \ell = 1, \\ \Theta(\ell^{\Lambda-1}) & \text{if } \ell > 1. \end{cases} \quad (\text{C.1b})$$

We take a total number of modes $m \geq R$ (even $m \gg R$ generally), and we take an input state $|\underline{n}\rangle = |1, 0, 1, 0, \dots\rangle$ that we used in Section 5. The analysis can be easily generalized to input $|\underline{n}\rangle = |n_0, n_1, \dots, n_{m-1}\rangle$ with $n_a \leq 1$ for all modes a , by bounding the photon numbers by R instead of $R/2$ in the following.

C.1 Time to compute transition amplitudes

C.1.1 Single beamsplitter

In our reference implementation, we use the following equation to compute the transition amplitudes for a beamsplitter $B_{ab}(\theta)$ between modes $a \neq b$ with beamsplitter angle θ :

where $l = N_a - k$, the number of photons transi-

tioning from b to a . Furthermore n_a, n_b are the input, and N_a, N_b the output numbers of photons.⁶

Assuming $\Theta(1)$ time arithmetic⁷, the complexity of computing one transition amplitude from (C.2) is just the number of terms in the summation, i.e. the number of possible values of $k \in \llbracket \max\{0, N_b - n_a\}, \min\{n_a, N_a\} \rrbracket$. We show the size of this interval in the table below:

	$N_b \leq n_a$	$N_b > n_a$
$n_a \leq N_a$	$n_a + 1$	$2n_a - N_b + 1$
$n_a > N_a$	$N_a + 1$	$n_a + N_a - N_b + 1$

Note that $\tilde{n} := n_a + n_b = N_a + N_b$, and $N_a = k + l$. It is clear that in each case the number can be bounded by $2\tilde{n} + 1$.

Simplification: beamsplitters see all To compute evolution of the state through a beamsplitter, we need to apply (C.2) for all possible values of n_a, n_b, N_a, N_b . We can exactly predict the constraints on these using the lattice path formalism from Section 4, however, this would be too complicated for an asymptotic analysis. Instead, we simplify and assume that *every beamsplitter of a component P_c may interact with all the photons in that component*. This will lead to an upper bound, because most beamsplitters will see much fewer photons. The number of photons in the component P_c is denoted $n(P_c)$. Thus we bound $\tilde{n} = n_a + n_b \leq n(P_c)$.

There are $\tilde{n} + 1$ possible way to distribute $\tilde{n}$ photons between N_a, N_b . Likewise, there are $\tilde{n} + 1$ ways to do this for the input numbers of photons n_a, n_b , but we shall count these in a later step when we reason about the size of the statevector to be evolved. Thus for each beamsplitter we need to compute up to $\tilde{n} + 1 \leq n(P_c) + 1$ amplitudes, and together with the above number of terms per amplitude, we get up to $(n(P_c) + 1) \cdot (2n(P_c) + 1)$ operations.

C.1.2 All beamsplitters in the circuit

We now count how many beamsplitters are contained in the circuit so that we can estimate the

⁶To get (C.2) from (2.2), represent photons in modes a, b by creation operators $\hat{a}_a^\dagger, \hat{a}_b^\dagger$ that transform as $\hat{a}_c^\dagger \mapsto \sum_{d \in \{a, b\}} B(\theta)_{dc} \hat{a}_d^\dagger$, and extend this to $(\hat{a}_a^\dagger)^{n_a} (\hat{a}_b^\dagger)^{n_b}$.

⁷including using $q! = \exp \ln q! = \exp \sum_{i=1}^q \log i$ to efficiently compute factorials, as well as precomputing the first several values of $\log q!$

time needed to compute the transformation of the state through them all. Each loop of length ℓ' generates a sequence of $m - \ell'$ beamsplitters: there is a beamsplitter starting at each mode, except the ℓ' last ones. For simplicity, we upper bound this by m , noting also that generally $m \gg R > \ell'$. There are Λ loops, so the number of beamsplitters in the circuit is bounded by $m\Lambda$.

Note that the reason we can count all beamsplitters in the circuit, instead of worrying about individual components, is a later simplification that assumes all components behave similarly.

Observe that the complexity depends on the number of photons in mode P_c . This complicates the analysis, because these numbers depend on the actual SPBS marginal probabilities. We will again simplify the analysis by taking an appropriate assumption. Note first that

$$n(P_{c+1}) = n(P_c) - n'_c + n_{c+1+R},$$

which corresponds to removing the n'_c measured photons coming from the previous component P_c , and adding n_{c+1+R} photons in the new input mode.

Simplification: number of photons roughly constant Without any more information, it is reasonable to assume that on average, the number of photons in a component will stay roughly the same, i.e. $n(P_{c+1}) \approx n(P_c)$. This takes into account that we add photons, but not in all new input modes, and we remove photons by measurement, but we cannot predict how many. At its core, this is an entropy argument: we cannot assume any more information.

With this simplification, the number of photons in each component becomes $n(P_c) \approx n(P_0) = \lceil R/2 \rceil \approx R/2$ (or generally $\leq R$ for an input with $n_a \leq 1$). Then the number of operations needed to compute amplitudes becomes the number of operations per beamsplitter multiplied by the number of beamsplitters (still ignoring the state space):

$$\begin{aligned} T^{\text{amp}} &= \mathcal{O} \left(m\Lambda \cdot \left(\frac{R}{2} + 1 \right) \cdot (R + 1) \right) \\ &= \mathcal{O}(m\Lambda R^2). \end{aligned} \quad (\text{C.3})$$

Recall again that $R = 1 + \Lambda$ if $\ell = 1$, and $R = \mathcal{O}(\ell^{\Lambda-1})$ if $\ell > 1$. Then the time to compute the amplitudes of beamsplitters needed to evolve the

state is:

$$T^{\text{amp}} = \begin{cases} \mathcal{O}(m\Lambda^3) & \text{if } \ell = 1, \\ \mathcal{O}(m\Lambda\ell^{2(\Lambda-1)}) & \text{if } \ell > 1. \end{cases} \quad (\text{C.4})$$

C.2 Effect of state in memory

Finally, we recall that so far we only computed the time to compute the transition amplitudes. In order to evolve the state vector, we have to multiply its coordinate components by these values, which essentially means iterating over these components.

The number of statevector components, i.e. the dimension of the reachable state space, at each step, is given by the lattice paths. However, this would again make asymptotic analysis difficult. Instead, we extend the simplification we had already taken before: each beamsplitter can see all the photons in a progressive component P_c , and so we further assume that the Fock space at the input and output of each beamsplitter is the same as the final space at the end of the progressive component. Having assumed that $n(P_c) \approx n(P_0) \approx R/2$, we can bound the dimension of the relevant space by:

$$\binom{R + n(P_0) - 1}{R} \approx \binom{\frac{3}{2}R - 1}{R}$$

We simplify this using Stirling's approximation

$$\ln x! = x \ln x - x + \mathcal{O}(x)$$

which gives

$$\binom{x}{qx} \sim x(-q \ln q - (1-q) \ln(1-q))$$

for a large x and $q \in (0, 1)$. For us $x = \frac{3}{2}R - 1$ and $q \rightarrow \frac{2}{3}$ as R grows, and thus

$$\binom{\frac{3}{2}R - 1}{R} = \mathcal{O}\left(\left(\frac{\sqrt{27}}{2}\right)^R\right) = \mathcal{O}(2.6^R) \quad (\text{C.5})$$

where we take an upper bound $\sqrt{27}/2 = 2.598\dots < 2.6$.

Finally, noting that the above is independent of the component P_c , we obtain a bound on the total running time by multiplying the above with T^{amp} . We recall again that $R = 1 + \Lambda$ if $\ell = 1$, and $R = \mathcal{O}(\ell^{\Lambda-1})$ if $\ell > 1$. Thus the running time is:

$$T = \begin{cases} \mathcal{O}(m\Lambda^3 2.6^\Lambda) & \text{if } \ell = 1, \\ \mathcal{O}(m\Lambda\ell^{2(\Lambda-1)} 2.6^{\ell^{\Lambda-1}}) & \text{if } \ell > 1. \end{cases} \quad (\text{C.6})$$

C.2.1 Measurement subsumed

In the end, we note that we did not mention the measurement step that occurs after every progressive component P_c . This consists of iterating over the components of the state vector, summing up the marginal probabilities of sampling a particular photon number using a hash table, and then taking a sample from the resulting distribution. This can be done in time $\mathcal{O}(n(P_c)2.6^R) = \mathcal{O}(R2.6^R)$ that is already subsumed by the bound in (C.6).

References

- [1] Scott Aaronson and Alex Arkhipov. “The computational complexity of linear optics”. In Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. [Page 333–342](#). STOC ’11 New York, NY, USA (2011). Association for Computing Machinery.
- [2] L. G. Valiant. “The complexity of computing the permanent”. [Theoretical Computer Science](#) **8**, 189–201 (1979).
- [3] Michael Reck, Anton Zeilinger, Herbert J. Bernstein, and Philip Bertani. “Experimental realization of any discrete unitary operator”. [Phys. Rev. Lett.](#) **73**, 58–61 (1994).
- [4] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, W. Steven Kolthammer, and Ian A. Walmsley. “Optimal design for universal multiport interferometers”. [Optica](#) **3**, 1460–1465 (2016).
- [5] Raúl García-Patrón, Jelmer J. Renema, and Valery Shchesnovich. “Simulating boson sampling in lossy architectures”. [Quantum](#) **3**, 169 (2019).
- [6] Lars S. Madsen, Fabian Laudenbach, Mohsen Falamarzi Askarani, Fabien Rortais, Trevor Vincent, Jacob F. F. Bulmer, Filippo M. Miatto, Leonhard Neuhaus, Lukas G. Helt, Matthew J. Collins, Adriana E. Lita, Thomas Gerrits, Sae Woo Nam, Varun D. Vaidya, Matteo Menotti, Ish Dhand, Zachary Vernon, Nicolás Quesada, and Jonathan Lavoie. “Quantum computational advantage with a programmable photonic processor”. [Nature](#) **606**, 75–81 (2022).

- [7] Keith R. Motes, Alexei Gilchrist, Jonathan P. Dowling, and Peter P. Rohde. “Scalable Boson Sampling with Time-Bin Encoding Using a Loop-Based Architecture”. *Physical Review Letters* **113**, 120501 (2014).
- [8] Michael Lubasch, Antonio A. Valido, Jelmer J. Renema, W. Steven Kolthammer, Dieter Jaksch, M. S. Kim, Ian Walmsley, and Raúl García-Patrón. “Tensor network states in time-bin quantum optics”. *Phys. Rev. A* **97**, 062304 (2018).
- [9] Abhinav Deshpande, Arthur Mehta, Trevor Vincent, Nicolás Quesada, Marcel Hinsche, Marios Ioannou, Lars Madsen, Jonathan Lavoie, Haoyu Qi, Jens Eisert, Dominik Hangleiter, Bill Fefferman, and Ish Dhand. “Quantum computational advantage via high-dimensional gaussian boson sampling”. *Science Advances* **8**, eabi7894 (2022).
- [10] Kamil Bradler and Hugo Wallner. “Certain properties and applications of shallow bosonic circuits” (2021). [arXiv:2112.09766 \[quant-ph\]](#).
- [11] Peter Clifford and Raphaël Clifford. “The classical complexity of boson sampling”. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Page 146–155. SODA ’18USA (2018). Society for Industrial and Applied Mathematics.
- [12] Changhun Oh, Youngrong Lim, Bill Fefferman, and Liang Jiang. “Classical Simulation of Boson Sampling Based on Graph Structure”. *Physical Review Letters* **128**, 190501 (2022).
- [13] Samo Novák and Raúl García-Patrón. “Laplace expansions and tree decompositions: polytime algorithm for shallow nearest-neighbour boson sampling” (2024). [arXiv:2412.18664 \[quant-ph\]](#).
- [14] Nicolas Heurtel, Shane Mansfield, Jean Senellart, and Benoît Valiron. “Strong Simulation of Linear Optical Processes”. *Computer Physics Communications* **291** (2023).
- [15] Changhun Oh, Minzhao Liu, Yuri Alexeev, Bill Fefferman, and Liang Jiang. “Classical algorithm for simulating experimental gaussian boson sampling”. *Nature Physics* **20**, 1461–1468 (2024).
- [16] Byeongseon Go, Changhun Oh, Liang Jiang, and Hyunseok Jeong. “Exploring shallow-depth boson sampling: Toward a scalable quantum advantage”. *Phys. Rev. A* **109**, 052613 (2024).
- [17] Byeongseon Go, Changhun Oh, and Hyunseok Jeong. “On computational complexity and average-case hardness of shallow-depth boson sampling” (2024). [arXiv:2405.01786 \[quant-ph\]](#).
- [18] Richard A. Campos, Bahaa E. A. Saleh, and Malvin C. Teich. “Quantum-mechanical lossless beam splitter: Su(2) symmetry and photon statistics”. *Phys. Rev. A* **40**, 1371–1384 (1989).
- [19] Lorenzo Carosini, Virginia Oddi, Francesco Giorgino, Lena M. Hansen, Benoit Seron, Simone Piacentini, Tobias Guggemos, Iris Agresti, Juan C. Loredó, and Philip Walther. “Programmable multiphoton quantum interference in a single spatial mode”. *Science Advances* **10** (2024).
- [20] Patrik I. Sund, Ravitej Uppu, Stefano Pavesani, and Peter Lodahl. “Hardware requirements for realizing a quantum advantage with deterministic single-photon sources”. *Phys. Rev. A* **109**, 042613 (2024).
- [21] Yu He, X. Ding, Z.-E. Su, H.-L. Huang, J. Qin, C. Wang, S. Unsleber, C. Chen, H. Wang, Y.-M. He, X.-L. Wang, W.-J. Zhang, S.-J. Chen, C. Schneider, M. Kamp, L.-X. You, Z. Wang, S. Höfling, Chao-Yang Lu, and Jian-Wei Pan. “Time-bin-encoded boson sampling with a single-photon device”. *Phys. Rev. Lett.* **118**, 190501 (2017).
- [22] Hui Wang, Jian Qin, Xing Ding, Ming-Cheng Chen, Si Chen, Xiang You, Yu-Ming He, Xiao Jiang, L. You, Z. Wang, C. Schneider, Jelmer J. Renema, Sven Höfling, Chao-Yang Lu, and Jian-Wei Pan. “Boson sampling with 20 input photons and a 60-mode interferometer in a 10^{14} -dimensional Hilbert space”. *Phys. Rev. Lett.* **123**, 250503 (2019).
- [23] Adam Bouland, Daniel Brod, Ishaun Datta, Bill Fefferman, Daniel Grier, Felipe Hernandez, and Michal Oszmaniec. “Complexity-theoretic foundations of BosonSampling

- with a linear number of modes” (2023). [arXiv:2312.00286](#) [quant-ph].
- [24] Berwin A. Turlach. “Bandwidth selection in kernel density estimation: a review” (1999). *Statistic und Oekonometrie* **9307**, Humboldt Universitaet Berlin.
- [25] “Frontier User Guide — OLCF User Documentation” (2024). https://docs.olcf.ornl.gov/systems/frontier_user_guide.html, accessed 2024-09-05.
- [26] J. J. Renema, A. Menssen, W. R. Clements, G. Triginer, W. S. Kolthammer, and I. A. Walmsley. “Efficient classical algorithm for boson sampling with partially distinguishable photons”. *Phys. Rev. Lett.* **120**, 220502 (2018).
- [27] Christopher Sparrow. “Quantum interference in universal linear optical devices for quantum computation and simulation”. *PhD thesis*. Imperial College London. (2018).
- [28] V. S. Shchesnovich. “Partial indistinguishability theory for multiphoton experiments in multiport devices”. *Phys. Rev. A* **91**, 013844 (2015).
- [29] Emilio Annoni and Stephen C Wein. “Incoherent behavior of partially distinguishable photons” (2025). [arXiv:2502.05047](#) [quant-ph].
- [30] J.C. Rosales and P.A. García-Sánchez. “Numerical semigroups”. *Developments in Mathematics*. Springer New York. (2009).
- [31] Richard P. Stanley. “Enumerative combinatorics”. *Number 49, 208 in Cambridge studies in advanced mathematics*. Cambridge University Press. Cambridge, NY (2012). Second edition.