Efficient Simulation of Shallow Quantum Circuits: Boson Sampling

Samo Novák

4th Year Project Report
Computer Science and Physics
School of Informatics
University of Edinburgh

2022

Abstract

In this work, we explore the classical simulation of a process called Boson Sampling. Here, we have an interferometer acting on m modes of a photon. We send n individual photons into it and measure where they come out. Boson Sampling is a model suggested by Aaronson and Arkhipov [2] as a method for experimental demonstration of quantum supremacy on near-term devices. Being able to simulate it classically raises the bar for quantum supremacy. The current best algorithm to generate samples for arbitrary interferometers is by Clifford & Clifford[11]. It creates the sample progressively, photon by photon, by sampling from the marginal distributions. Their algorithm has running time $O(n2^n + \mathrm{poly}(m,n))$, where $\mathrm{poly}(m,n) = O(mn^2)$.

The most significant part of the complexity of Boson Sampling comes from the need to compute permanents of matrices; performing this computation for an arbitrary matrix is #P-hard.[1] However, the situation is different if we can identify some additional structure. In this work, we focus on sampling from a shallow interferometer, i.e. one that has depth $D = O(\log m)$. We give a construction of such a device and prove that this induces a band on the matrix. We use the bipartite tree decomposition method of Cifuentes & Parrilo from [9], which allows us to compute the required permanents efficiently.

We use this to adapt the algorithms by Clifford & Clifford, replacing all permanent computations with this tree-decomposition-based method. We give an algorithm that generates a sample from the collision-less shallow Boson Sampling experiment in time $O\big(mn^3 2^{\omega'(n)}\omega'(n)^2\big)$ where $\omega'(n) \coloneqq O(\min(c\log m, 2n))$; or for $m = n^2$ in time $O\left(n^{5+c}c^2\log^2 n\right)$. The constant c is a parameter that we can choose: we introduce it into the depth: $D \coloneqq O(c\log m)$. Our algorithm first constructs a tree decomposition of the randomly permuted first n columns of the interferometer matrix and then uses restrictions of the decomposition (a concept we introduce) to compute permanents of the required submatrices.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Samo Novák*)

Acknowledgements

Firstly, I would like to mention my supervisor Raúl García-Patrón. His help and guidance were indispensable. Our discussions were always fruitful and moved me forward with feedback and motivation. Thank you.

I want to thank my family, in particular my parents. They were an unending source of support on my university journey, including this project, and none of it would be possible without them.

Last but definitely not least, I would like to thank all my friends for always supporting and listening to me, be it remotely or in person. Some of you probably dreamt about graphs and permanents at some point.

Table of Contents

Chapter 1

Introduction

Quantum Computing is a promising field at the intersection of Computer Science and Physics. It exploits properties of quantum mechanical systems that do not have non-quantum (usually called *classical*) analogue to speed up computation. Some of these properties are superposition, interference and entanglement.[21]

A few examples of problems where quantum computation is better than classical include: simulation of quantum systems, optimization, and unstructured search (Grover's algorithm[17]). A major example is Shor's algorithm for integer factorization[24], which can use quantum properties to perform, in polynomial time, computation widely believed infeasible on classical computers. The hardness of integer factorization is the basis of many current cryptosystems (e.g. RSA), and Shor's algorithm would make quantum computers extremely efficient tools for cryptanalysis. Other use cases are being researched, such as quantum machine learning.[23]

In terms of theory, it is clear that quantum computers should be able to solve certain problems much more efficiently than classical. This is what we call *quantum advantage* or *supremacy*. However, this is difficult to demonstrate experimentally: current quantum devices are limited in size, that is, the number of quantum bits (*qubits*) on which we can operate, and they are extremely delicate. Near-term quantum devices are not *fault-tolerant*, which means that it is difficult to use them for any elaborate computation that could demonstrate quantum advantage in an experiment.

More and more experiments with bigger or more complex systems are successfully being done to show the advantage. However, the bar for quantum supremacy will rise if we can show a classical algorithm that efficiently can perform the same computation, or simulate the quantum algorithm. This is the central motivation for this project.

We focus on the problem called *Boson Sampling*, defined by Aaronson & Arkhipov in [2]. Imagine we have an optical device called an *interferometer*, into which we are sending individual *photons*, particles of light. The incoming photons are in pre-defined *modes*, which could mean, for example, their locations in space. The interferometer scatters them: it sends photons to some output modes, which may be the same or different as input modes. We measure where the photons end up. This is a quantum mechanical process which is interesting because it can be used as a non-universal quantum computer

for certain specialised computations. Boson Sampling is the problem of simulating this process of photons scattering in a given interferometer, i.e. creating samples that obey the required probability distribution.

Generating these samples is a difficult problem because the probability distribution governing into which output modes the photons should go depends on an object called the *permanent* of a matrix. This is similar to a determinant, but does not alternate signs. However, while determinants are easy to compute, the computation of a permanent of an arbitrary matrix is #P-hard.[1] The best-known algorithms to compute a permanent take time $O(n2^n)$.[16]

In [20], Moylett & Turner showed how we can use a quantum computer to generate samples from Boson Sampling. In [11], Cliffords & Clifford gave two *classical* algorithms and proved that samples of the Boson Sampling experiment can be generated in $O(n2^n + \text{poly}(m,n))$ time (up to a constant factor as fast as computing two permanents) and $O(m)$ additional space; with m being the number of modes, and n the number of photons.

We focus on the case where the circuit (the interferometer) is *shallow*, meaning that its depth scales as $O(\log m)$, where m is the number of modes. There are some results about shallow circuits: in particular, in [18], Jozsa proved that they should be classically simulatable in polynomial time. In [15], García-Patrón, Renema & Shchesnovich showed that in a shallow regime, samples can be generates in quasi-polynomial time.

The matrix representing a shallow interferometer is *sparse*, which allows us to use an algorithm by Cifuentes & Parrilo[9] that can compute the permanent of an $n \times n$ sparse matrix using a technique called *bipartite tree decomposition* in time $\widetilde{O}(n2^{\omega})$, where ω is the *treewidth* of the decomposition, and $\widetilde{O}$ hides polynomial factors of ω.

In this work, we merge the methods by Clifford & Clifford for classical simulation of Boson Sampling, and Cifuentes & Parrilo for efficient computation of permanents of sparse matrices. We derive a polynomial-time algorithm to generate samples of the Boson Sampling experiment for shallow interferometers. We explore the underlying concepts of Boson Sampling, quantum optics as well as the mathematics of tree decompositions.

The problem we are solving seems to be of broader research interest. During the work on this project, Oh et al. published a pre-print[22] that described a similar idea. However, they focused on other problems (especially Gaussian Boson Sampling), and it seems they do not give much detail about a solution to our problem. To the best of our understanding, our approach is different in that we generate a large tree decomposition of the interferometer matrix and then use its restrictions (see sec. 5.2) to compute permanents.

1.1 Aims

1. Understand Boson Sampling and the algorithms of Clifford & Clifford[11] for generating samples from arbitrary interferometers,

2. understand tree decompositions and the algorithm due to Cifuentes & Parrilo[9] that can compute permanents of sparse matrices efficiently,
3. exploit the previous two points to obtain a polynomial-time classical algorithm that can simulate Boson Sampling on shallow interferometers.

1.2 Structure

In chapter 2, we present the basic mathematics and quantum physics needed to understand the work. Then chapters 3 and 4 go on to explain the two main ideas: Boson Sampling and the Clifford & Clifford paper; and the algorithm of Cifuentes & Parrilo. These are highly non-trivial and took a lot of work to understand; hence we give them their own chapters. In chapter 5, we bring the knowledge gained from them, as well as original ideas, together and give our own algorithm to simulate Boson Sampling. Finally, in chapter 6, we conclude, reflect on the results, and leave two conjectures for the future.

Additionally, for the interested reader, some further ideas are sketched in the appendices: these are not central to the project and not necessary to understand it. The topics shown include alternative tree decomposition methods that we did not use.

1.3 Contributions

Here, we summarise the main contributions from this work:

- We define a concept of restrictions of tree decompositions (see definition 5.2.1), which removes unwanted vertices from an existing decomposition. If the original matrix has structure that gives its graph an easy decomposition (such as a banded matrix), an arbitrary submatrix may be harder to decompose. Our method helps by leveraging the decomposition of the original matrix. We also prove a result relating this to the permanent (see lemma 5.2.3).
- We propose a new algorithm to simulate Boson Sampling on shallow circuits in time $O\big(mn^3 2^{\omega'(n)}\omega'(n)^2\big)$ where $\omega'(n) := O(\min(c\log m, 2n))$; or for $m = n^2$ in time $O\left(n^{5+c}c^2\log^2 n\right)$. We can choose the constant c depending on the desired depth $D := O(c\log m)$.

  Our algorithm works by progressively sampling photons, using the algorithm of Clifford & Clifford. The innovation here is that before sampling, we generate a tree decomposition of the interferometer matrix, and then replace the permanent computation with the algorithm of Cifuentes & Parrilo. We can use the overall decomposition and, at each step, restrict it to only the relevant parts. This enables us to compute permanents of submatrices efficiently. See section 5.3.
- We extend the bra-ket notation to graphs of matrices; this is our own spin on some usual notations, which we think is useful (see notation 2.3.1).

Chapter 2

General Preliminaries

Firstly, we present a basic table of symbols. These are standard ideas, but we also take this opportunity to fix their notation. We will add more along the way.

symbol	meaning	definition
$\mathbb{Z}$	integers	$=\{...,-2,-1,0,1,2,...\}$
$\mathbb{Z}_+$	positive integers	$=\{1,2,...\}$
$[n]$	positive integers up to $n \in \mathbb{Z}_+$	$=\{1,2,...,n\}$
$\mathbb{N}$	natural numbers	$=\{0\} \cup \mathbb{Z}_+$
$\mathbb{C}$	complex numbers	
z^*	complex conjugate of $z \in \mathbb{C}$	
$O^\dagger$	conjugate transpose of operator O	$=\left(O^\top\right)^*$
$\mathbb{U}(m)$	unitary operators of dimension m	
$\lvert X\rvert$	cardinality of set X	
$\mathbb{P}[\varphi]$	probability of φ	
pmf	probability mass function	
$Bij(X,Y)$	all bijections from X to Y	$=\{f : X \to Y \mid f \text{ is a bijection}\}$
S_X	symmetric group on X	$=Bij(X,X)$
S_n	symmetric group on $n \in \mathbb{Z}_+$ elements	$=\mathrm{S}_{[n]}$
$X \sqcup Y$	disjoint union of sets X,Y	
$\mathcal{P}(X)$	power set of X	$=\{Y \mid Y \subseteq X\}$

2.1 Dirac Notation

The Dirac notation[13], also called *bra-ket* notation, is used ubiquitously in Quantum Theory, and we will be using it in this work as well. We will largely assume this is known, but we recapitulate the relevant basics.

Notation 2.1.1 (bra-ket)**.** A quantum state is a vector in an abstract Hilbert space, denoted by a *ket* $|\varphi\rangle$, where φ is just the label. Its adjoint is called a *bra* $\langle\varphi|$. An *inner product* (a scalar) is denoted $\langle\psi|\varphi\rangle$, and an *outer product* (an operator) is denoted $|\psi\rangle\langle\varphi|$. A nice property of this notation is how these objects compose: the bras connect to kets

that are written to the right of them, e.g.:

$$|a\rangle\langle b| \cdot \left(\frac{1}{\sqrt{2}} |c\rangle + \frac{1}{\sqrt{2}} |d\rangle \right) = \frac{1}{\sqrt{2}} |a\rangle \left(\langle b|c\rangle + \langle b|d\rangle \right)$$

We can write any linear transformation $\hat{O}$ as a linear superposition of ket-bra outer products, provided we sum over an appropriate (orthonormal) basis:

$$\hat{O} = \left(\sum_i |i\rangle\langle i| \right) \hat{O} \left(\sum_j |j\rangle\langle j| \right) = \sum_i \sum_j O_{i,j} |i\rangle\langle j| \tag{2.1}$$

where $O_{i,j} := \langle i| \hat{O} |j\rangle$ are *matrix elements* of $\hat{O}$ in this basis. We clarify this term in the next section.

2.2 Matrices and Their Graphs

We will be working with matrices and their parts throughout the entire work, and we will use a formally slightly more general definition than usual. This corresponds to definitions that Cifuentes and Parrilo use in [9].

Definition 2.2.1 (matrix)**.** Let $\mathbb{F}$ be a field. Let $\mathcal{R}$ be a set of row labels, and let $\mathcal{C}$ be the set of column labels. The sets $\mathcal{R}$ and $\mathcal{C}$ are usually required to be finite, and this will always be the case for us. A *matrix* associates values in $\mathbb{F}$ to pairs of row and column labels:

$$\begin{aligned} M \;:\; \mathcal{R} \times \mathcal{C} &\to \mathbb{F} \\ (i,j) &\mapsto M_{i,j} \quad \forall i \in \mathcal{R}, j \in \mathcal{C} \end{aligned}$$

where $M_{i,j}$ are the *entries* or *matrix elements*. This is also denoted as $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$.

We used the same notation and name *matrix element* for an entry of a matrix, and in section 2.1 for $\langle i| \hat{O} |j\rangle$. This is no coincidence: for a fixed basis, an operator can be identified with a matrix of its matrix elements.

Notation 2.2.2. If $\mathcal{R} = [m]$ and $\mathcal{C} = [n]$ for some $m, n \in \mathbb{Z}_+$, then we denote the matrix as $M \in \mathbb{F}^{m\times n} := \mathbb{F}^{[m]\times[n]}$. This is a standard simplification.

Definition 2.2.3 (submatrix)**.** Let $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$ be a matrix. Let $\mathcal{R}' \subseteq \mathcal{R}$, $\mathcal{C}' \subseteq \mathcal{C}$ be subsets of row and column labels. Then the *submatrix* $M\big|_{\mathcal{R}',\mathcal{C}'}$ is just a domain restriction of M:

$$\begin{aligned} M\big|_{\mathcal{R}',\mathcal{C}'} \;:\; \mathcal{R}' \times \mathcal{C}' &\to \mathbb{F} \\ (i,j) &\mapsto M_{i,j} \end{aligned}$$

This is exactly what you would expect a submatrix to be: take a matrix and delete some rows and columns. Note that it is important that we do *not* relabel rows and columns to fill in the gaps:

$$\begin{array}{c} \\ 1 \\ \mathbf{2} \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{\mathbf{3},1} & M_{\mathbf{3},2} \end{pmatrix} \end{array} \neq \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{pmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{pmatrix} \end{array}\Bigg|_{\substack{\mathcal{R}'=\{1,\mathbf{3}\} \\ \mathcal{C}'=\{1,2\}}} = \begin{array}{c} \\ 1 \\ \mathbf{3} \end{array}\!\!\begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{pmatrix} M_{1,1} & M_{1,2} \\ M_{\mathbf{3},1} & M_{\mathbf{3},2} \end{pmatrix} \end{array}$$

An equivalent way to look at a matrix, which will become indispensable in chapter 4, is by representing it as a graph:

Definition 2.2.4 (graph of a matrix)**.** Let $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$ be a matrix. Then we define a weighted bipartite[1] graph $G(M) := (V, E, \ell)$ with:

- vertex set $V = \mathcal{R} \sqcup \mathcal{C}$ (disjoint union),
- edge set $E \subseteq \mathcal{R} \times \mathcal{C}$,
- labels, or weights, on edges $\ell : E \to \mathbb{F}$,

such that there exists an edge $(r,c) \in E$ iff the matrix element that the endpoints label is nonzero: $M_{r,c} \neq 0$. The function ℓ gives us the value of that matrix element, i.e. for all $(r,c) \in E$, we have $\ell(r,c) := M_{r,c}$.

A matrix and its graph uniquely determine each other, so we may view them as interchangeable. In fact, in chapter 4, we will use these graphs to compute permanents; and in chapter 5, we will use the graphs to reason about constructing matrices with properties we like.

2.3 Extending Dirac Notation to Graphs of Matrices

Here, we present our original notation for matrix graphs. It is similar to some other standard notations,[2] with a few key differences that arose naturally from our concrete needs. We look back at equation 2.1, but rebracket as follows

$$\hat{O} = \sum_i |i\rangle \langle i| \hat{O} \sum_j |j\rangle \langle j| = \sum_i \sum_j \Big(|i\rangle\Big)\Big(\langle i| \hat{O} |j\rangle\Big)\Big(\langle j|\Big) \tag{2.2}$$

It is useful to think of the expression on the right-hand side in 2.2 as an outer product of $|i\rangle$ and $\langle j|$, that are *connected* by the scalar $\langle i| \hat{O} |j\rangle$. We use this to represent the matrix as a graph:

Notation 2.3.1 (bra-ket notation for the graph of a matrix)**.** In the graph representation of definition 2.2.4, the row labels in $\mathcal{R}$ are denoted using kets $|i\rangle$, and the column labels in $\mathcal{C}$ by bras $\langle j|$. An edge represents a nonzero $\langle i| \hat{O} |j\rangle = O_{i,j}$.

[1] all our graphs will be bipartite, so we will sometimes omit this word

[2] especially for relations

Example 2.3.2. Let $M \in \mathbb{C}^{3\times 4}$ and $N \in \mathbb{C}^{4\times 2}$ be

$$M = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 4 & 5 \end{pmatrix} \qquad N = \begin{pmatrix} 1 & 5 \\ 0 & 7 \\ 0 & 2 \\ -1 & 0 \end{pmatrix} \tag{2.3}$$

Then the graphs $G(M)$ and $G(N)$ are:

$$G(M) = \left[\begin{array}{ll} |1\rangle \xrightarrow{\langle 1|M|1\rangle = 1} \langle 1| & \\ |2\rangle \xrightarrow{3} \langle 2| & \\ |3\rangle \xrightarrow{4} \langle 3| & \\ \langle 3|M|4\rangle = 5 \quad \langle 4| & \end{array}\right] \qquad G(N) = \left[\begin{array}{ll} |1\rangle \xrightarrow{1} \langle 1| & \\ |2\rangle \xrightarrow{7} \langle 2| & \\ |3\rangle & \\ |4\rangle & \langle 4|N|1\rangle = -1 \end{array}\right] \tag{2.4}$$

Notice how this notation is purposefully internal: we look at the matrix from the inside. We use this to construct matrices in section 5.1, so composition has to be immediate: clip together the bras and kets, sum over possible composite edges from $|i\rangle$ to $\langle j|$, multiply the weights of their parts. This is to mimic the usual composition of matrices.

Example 2.3.3. We use the same matrices M,N from example 2.3.2. Note especially how the edge with weight $1\cdot 5 + 2\cdot 7$ arises from the edges drawn bold.

$$G(MN) = \left[\begin{array}{ll} |1\rangle \xrightarrow{1} \langle 1| & \\ |2\rangle \xrightarrow{3} \langle 2| & \\ |3\rangle \xrightarrow{4} \langle 3| & \\ & \xrightarrow{5} \langle 4| \end{array}\right] \cdot \left[\begin{array}{ll} |1\rangle \xrightarrow{1} \langle 1| & \\ |2\rangle \xrightarrow{7} \langle 2| & \\ |3\rangle & \\ |4\rangle & \xrightarrow{-1} \end{array}\right] \tag{2.5}$$

$$= \left[\begin{array}{lll} |1\rangle \xrightarrow{1} & \langle 1|1\rangle & \xrightarrow{1} \langle 1| \\ |2\rangle \xrightarrow{3} & \langle 2|2\rangle & \xrightarrow{7} \langle 2| \\ |3\rangle \xrightarrow{4} & \langle 3|3\rangle & \xrightarrow{2} \\ & \langle 4|4\rangle & \xrightarrow{-1} \end{array}\right] = \left[\begin{array}{ll} |1\rangle \xrightarrow{1\cdot 1} & \langle 1| \\ & \xrightarrow{1\cdot 5+2\cdot 7} \\ |2\rangle \xrightarrow{3\cdot 1} & \langle 2| \\ |3\rangle \xrightarrow{4\cdot 2} & \\ & \xrightarrow{5\cdot(-1)} \end{array}\right] \tag{2.6}$$

The multiplication is linear, but we only show those parts that do not vanish, i.e. $\langle i|i\rangle$. We emphasize that this is meant to faithfully represent composition, so data is seen as flowing from *right to left*: any vector $|v\rangle$ to which MN is applied comes from right, same as in the usual notation $MN|v\rangle$. Likewise, bras are applied from left; for example:

$$\langle i|MN|j\rangle = \langle i| \left[\begin{array}{ll} |1\rangle \xrightarrow{1\cdot 1} & \langle 1| \\ & \xrightarrow{1\cdot 5+2\cdot 7} \\ |2\rangle \xrightarrow{3\cdot 1} & \langle 2| \\ |3\rangle \xrightarrow{4\cdot 2} & \\ & \xrightarrow{5\cdot(-1)} \end{array}\right] |j\rangle \tag{2.7}$$

2.4 Permanent

The essential, main aim of this work is to be able to compute the permanents of interesting matrices. Let us define what a permanent is first then.

Definition 2.4.1 (permanent of a matrix; c.f. [16])**.** Let $M \in \mathbb{F}^{n\times n}$ be a square matrix. Then the permanent of this matrix is

$$\operatorname{per} M := \sum_{\sigma\in S_n} \prod_{i=1}^{n} M_{i,\sigma(i)}$$

where S_n is the symmetric group of n elements.

The above is the standard definition. However, for our needs it is better to redefine this to correspond with our definition of a matrix (def. 2.2.1):

Definition 2.4.2. For a matrix $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$, its permanent is

$$\operatorname{per} M := \sum_{\pi\in Bij(\mathcal{R},\mathcal{C})} \prod_{r\in\mathcal{R}} M_{r,\pi(r)}.$$

Recall that $Bij(\mathcal{R},\mathcal{C})$ is the set of bijections $\mathcal{R}\to\mathcal{C}$.

We do not require a square matrix anymore. When examining Cifuentes and Parrilo's [9], we will need permanents of submatrices that are not square; their permanents are defined to be zero. In addition, we will need the permanent of a matrix without any rows or columns, which they implicitly assumed to be one.[3] In fact, definition 2.4.2 is flexible enough to handle both of these cases:

Lemma 2.4.3 (arbitrary permanents)**.** *Let $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$ be a matrix. If $|\mathcal{R}| \neq |\mathcal{C}|$, then* $\operatorname{per} M = 0$*; and if $\mathcal{R} = \mathcal{C} = \varnothing$, then* $\operatorname{per} M = 1$.

Proof. We work with definition 2.4.2. In the case $|\mathcal{R}| \neq |\mathcal{C}|$, the set of bijections $Bij(\mathcal{R},\mathcal{C})$ is empty – no bijections are possible between sets of different cardinality. We sum over an empty set, which gives us $\sum_{\pi\in\varnothing}(\cdots) = 0$, the additive unit. Hence $\operatorname{per} M = 0$.

In the case $\mathcal{R} = \mathcal{C} = \varnothing$, there is a single bijection, the empty function $\varepsilon : \varnothing \to \varnothing$. The sum is over the singleton set $Bij(\mathcal{R},\mathcal{C}) = \{\varepsilon\}$, and inside we take a product over an empty set $\mathcal{R}$:

$$\sum_{\pi\in\{\varepsilon\}} \prod_{r\in\varnothing} M_{r,\pi(r)} = \prod_{r\in\varnothing} M_{r,\varepsilon(r)} = 1.$$

Note that a product of zero elements is one. This is the same idea as factorial $0! = 1$. Hence $\operatorname{per} M = 1$. ∎

2.4.1 Permanents Sum Perfect Matchings

The graph representation of a matrix connects naturally to permanents.[19, 9] So far, we had a matrix and we used a graph to represent it. Now we go the other way, from a graph to the matrix. and we introduce the intuition using figure 2.1.

[3] this case caused much confusion when trying to understand the paper, so we clarify it beforehand

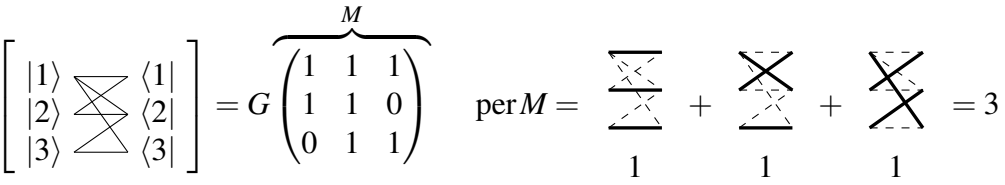


Figure 2.1: Connection between permanents and graphs of matrices

The matrix M is exactly the adjacency matrix of a directed bipartite graph $G(M)$ containing weights of the edges. The permanent counts all the perfect matchings, i.e. bijections $\mathcal{R} \to \mathcal{C}$ in this graph. If the edges have weights, then this is a weighted summation, where terms are products of weights of the edges in a matching.

Lemma 2.4.4. *Let $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$ be a matrix. Define $\mathfrak{M}_M(\mathcal{R},\mathcal{C}) \coloneqq \{\pi \in Bij(\mathcal{R},\mathcal{C}) \mid \forall r \in \mathcal{R}.\ M_{r,\pi(r)} \neq 0\}$ as the set of bijections that do not allow any zero matrix elements, or equivalently any zero edges in $G(M)$. Then*

$$\operatorname{per} M = \sum_{\pi\in\mathfrak{M}_M(\mathcal{R},\mathcal{C})} \prod_{r\in\mathcal{R}} M_{r,\pi(r)}$$

Proof. We start with a sum over all bijections. However, any $\pi \in Bij(\mathcal{R},\mathcal{C})$ that allows a matrix element $M_{r,\pi(r)} = 0$ will make the whole product vanish. Then only those summands remain that do not allow any $M_{r,\pi(r)}$ to be zero; i.e. exactly $\mathfrak{M}_M(\mathcal{R},\mathcal{C})$. ■

The definition of $\mathfrak{M}_M(\mathcal{R},\mathcal{C})$ captures exactly the perfect matchings in the graph $G(M)$. Some properties of permanents are immediate if we reason about graphs and matchings. The one we will care about most is the following result:

Lemma 2.4.5. *The permanent is invariant under permutations of rows or columns.*

Proof. (cf. Theorem 1.1 (b) in [19]) Let $M \in \mathbb{F}^{\mathcal{R}\times\mathcal{C}}$ be a matrix, and let $\sigma \in \mathrm{S}_{\mathcal{R}}$ and $\tau \in \mathrm{S}_{\mathcal{C}}$ be permutations of rows and columns respectively. We can express $\sigma = \sum_{r\in\mathcal{R}} |\sigma(r)\rangle\langle r|$ and $\tau = \sum_{c\in\mathcal{C}} |c\rangle\langle\tau(c)|$. When composed with M from left (σM) or right ($M\tau$), these just rename vertices, but do not change their connectivity or edge weights. Hence, the permanent, a sum over matchings, must be the same. ■

2.4.2 Complexity of Computing the Permanent

The currently best known algorithms to compute a permanent of some arbitrary matrix $M \in \mathbb{F}^{n\times n}$ require $O(n2^n)$ operations. We give here two such examples: Ryser's formula and Glynn's formula[16]; we will not use them, but we are interested in the comparisons.

$$\text{Ryser:}\quad \operatorname{per} M = \sum_{S\subseteq[n]} (-1)^{|S|} \prod_{j=1}^{n} \sum_{i\in[n]\setminus S} M_{i,j} \tag{2.8}$$

$$\text{Glynn:}\quad \operatorname{per} M = \frac{1}{2^{n-1}} \left[\sum_{\substack{\underline{\delta}\in\{\pm1\}^n \\ \delta_1=+1}} \left(\prod_{a=1}^{n} \delta_a \right) \prod_{c=1}^{n} \sum_{b=1}^{n} \delta_b M_{b,c} \right] \tag{2.9}$$

In Ryser's formula (2.8), we iterate over $S \subseteq [n]$ in Gray code order. Glynn's formula (2.9) requires that the characteristis of $\mathbb{F}$ is not two.[16]

2.5 Short Introduction of Quantum Optics

In quantum optics, we are dealing with *photons*, particles of light, existing in *modes*. These could just be locations, but also for example polarisations. We have two representations used to talk about these systems: first the *number states*, also called *Fock states*. For some mode, these count how many photons exist in that mode, and we denote them as $|n\rangle$, where we have $n \in \mathbb{N}$ photons. Canonically, such a state is written as an action of the *creation operator* $\hat{a}^\dagger$ acting on a *vacuum state* $|0\rangle$: [14]

$$|n\rangle = \frac{(\hat{a}^\dagger)^n}{\sqrt{n!}} |0\rangle \tag{2.10}$$

where the $\sqrt{n!}$ is normalisation, and the operator used is defined as follows:

Definition 2.5.1. *Creation* ($\hat{a}^\dagger$) and its corresponding *annihilation* ($\hat{a}$) operators act on number states and are defined for all $n \in \mathbb{N}$ as:

$$\hat{a}^\dagger |n\rangle = \sqrt{n+1}\,|n+1\rangle \qquad \hat{a}\,|n\rangle = \begin{cases} \sqrt{n}\,|n-1\rangle & n > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

These number states are eigenstates of the *number operator* $\hat{a}^\dagger\hat{a}$, satisfying $\hat{a}^\dagger\hat{a}\,|n\rangle = n\,|n\rangle$, i.e. a state (=vector) $|n\rangle$ is an eigenstate (=eigenvector) of this operator.

In we have m modes, then the number states are $|\underline{n}\rangle := |n_1, \ldots, n_m\rangle$, with the total number of photons $n = \sum_{i=1}^m n_i$. Expressed using creation operators:

$$|\underline{n}\rangle = \prod_{i=1}^m \frac{(\hat{a}_i^\dagger)^{n_i}}{\sqrt{n_i!}} |0\rangle^{\otimes m} \tag{2.12}$$

where $|0\rangle^{\otimes m} = |0\rangle_1 \otimes |0\rangle_2 \otimes \cdots \otimes |0\rangle_m$ is the tensor product of m vaccum states, and operator $\hat{a}_i^\dagger$ acts *only* on mode i. A linear optical device, such as *interferometer* (see sec. 2.5.3), represented by a matrix $\mathcal{U}$, acts on modes. It moves a photon between the modes. Following (2.12), we can express this as the action on the creation operators:

$$\hat{a}_i^\dagger \overset{\mathcal{U}}{\longmapsto} \sum_{j=1}^m \mathcal{U}_{i,j}\hat{a}_j^\dagger \tag{2.13}$$

In [3, chapter 3.2], Aaroronson and Arkhipov formalize this view that we may describe a state as an action of creation operators: states are represented by multivariate complex polynomials $\mathbb{C}[\hat{a}_1^\dagger, \hat{a}_2^\dagger, \ldots, \hat{a}_m^\dagger]$, where $\hat{a}_i^\dagger$ are now just formal variables. Transformations of these states are substitutions of the variables and work exactly as in (2.13).

2.5.1 Qudit Representation

Instead of number states which tell us the number of photons in each mode, we may work with individual photons, and ask in which modes they are. These will be our *qudits*. For example, if we have $n = 1$ photon in m possible modes, then a photon mode eigenstate[4] $|\psi\rangle \in \mathbb{C}^m$ means this photon is in mode $\psi \in [m]$. However, photons are identical particles, and we can never tell which is which; hence with multiple photons in number state $|\underline{n}\rangle$, we must always have a symmetric superposition (cf. [20]):

$$|\underline{n}\rangle \simeq \frac{1}{\sqrt{n!\prod_{i=1}^{m} n_i!}} \sum_{\sigma \in \mathrm{S}_n} \sigma \left|\Psi_{\underline{n}}\right\rangle \tag{2.14}$$

where we denote $\sigma \in \mathrm{S}_n$ permutations which relabel the photons to give all reorderings, and

$$\left|\Psi_{\underline{n}}\right\rangle := |1\rangle^{\otimes n_1} \otimes |2\rangle^{\otimes n_2} \otimes \cdots \otimes |m\rangle^{\otimes n_m} = \bigotimes_{i=1}^{m} |i\rangle^{\otimes n_i} \tag{2.15}$$

is a qudit state in a non-decreasing ordering of the photon modes. We use the symbol $\simeq$ to mean these states are isomorphic: they represent the same physical system, but formally live in different spaces: $|\underline{n}\rangle \in (\mathbb{C}^n)^{\otimes m}$ and $\sum_\sigma \sigma \left|\Psi_{\underline{n}}\right\rangle \in (\mathbb{C}^m)^{\otimes n}$.

We do not give the derivation of the normalisation constant here. The proof is similar to the derivation in section 3.1, and the idea is that if we have n_i photons in some mode i, then any permutation which only relabels these, while keeping them all in mode i, actually gives the exact same state; and there are $n_i!$ of these permuations.

2.5.2 Transformation of Multiple Photons

In (2.13), we gave the action of an optical device acting on modes. Even if we have multiple photons, the transformation is clear (especially with the polynomial interpretation by Aaronson & Arkhipov[3]). How does it translate to the qudit representation?

In the qudit representation, the basis is formed by tensor products of photons that tell us in which mode the photon is: $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$. Then the transformation, acting on modes, will transform each factor of the product separately, actings as $\mathcal{U}^{\otimes n}$:

$$\text{modes: } \hat{a}_i^\dagger \overset{\mathcal{U}}{\longmapsto} \sum_{j=1}^{m} \mathcal{U}_{i,j} \hat{a}_j^\dagger \qquad \text{1 qudit: } |i\rangle \overset{\mathcal{U}}{\longmapsto} \sum_{j=1}^{m} \mathcal{U}_{i,j} |j\rangle = \mathcal{U}|i\rangle \tag{2.16}$$

$$n \text{ qudits: } \left|\Psi_{\underline{n}}\right\rangle = \bigotimes_{k=1}^{n} |\psi_k\rangle \overset{\mathcal{U}}{\longmapsto} \bigotimes_{k=1}^{n} \left(\mathcal{U}|\psi_k\rangle\right) = \mathcal{U}^{\otimes n} \left|\Psi_{\underline{n}}\right\rangle \tag{2.17}$$

2.5.3 Basic Optical Devices

We have mentioned the word *interferometer* before; here we give details: an interferometer is an optical system that splits photon beams (or, in our case, individual photons), sends them via different paths, and recombines them later. Along the way,

[4] this is an eigenstate of the mode operator $\hat{m}|\psi\rangle := \psi|\psi\rangle, \psi \in [m]$

they may acquire phases, and they interfere when they are combined. Importantly, they may interfere constructively, but also destructively: the probability amplitudes may cancel. Unless we involve losses of photons, such a device acts as a linear unitary transformation of modes; and even lossy devices can be decomposed into a series of unitary transformations and lossy channels.[15]

These interferometers may be constructed from simpler devices called *phase shifters* and *beamsplitters*. Any interferometer can be constructed by using only a combination of these two kinds of devices[8, 4], an important fact which we use in sec. 5.1.1. Let us define these devices:

Definition 2.5.2. A *phase gate* (or phase *shifter*), parametrised by an angle $\phi \in [0, 2\pi)$, acts on the mode i as follows

$$\text{modes:} \quad \hat{a}_i^\dagger \mapsto e^{i\phi}\, \hat{a}_i^\dagger \qquad\qquad \text{qudits:} \quad |i\rangle \xleftarrow{\;e^{i\phi}\;} \langle i| \tag{2.18}$$

The arrow in the graphical representation empasises that we see this as photons in mode i going from right to left. Phase shifters are single-mode unitary operations; from now on, to simplify, we will always multiply them into the preceding or following operations, as suggested by Jozsa in [18].

The other basic building block is a beamsplitter, which is an optical element that reflects or transmits a photon with some probability, i.e. it may change the mode of a photon.

Definition 2.5.3 (cf. [7])**.** Let $\theta, \phi_R, \phi_T \in [0, 2\pi)$ be three angles. The beamsplitter acting on modes i, j parametrised by these angles is:

$$B(\theta, \phi_T, \phi_R) = \begin{pmatrix} e^{i\phi_T}\cos\theta & e^{i\phi_R}\sin\theta \\ -e^{-i\phi_R}\sin\theta & e^{-i\phi_T}\cos\theta \end{pmatrix} \simeq \left[\begin{array}{cc} |i\rangle \xrightarrow{\;e^{i\phi_T}\cos\theta\;} \langle i| & \\ & \quad -e^{-i\phi_R}\sin\theta \\ |j\rangle \xrightarrow{\;e^{-i\phi_T}\cos\theta\;} \langle j| & \\ & \quad e^{i\phi_R}\sin\theta \end{array}\right] \tag{2.19}$$

where $\phi_{T\ (\text{resp. } R)}$ are phases applied to the transmitted (resp. reflected) photon. The probability of transmission is $\cos^2\theta$, and of reflection $\sin^2\theta$; the angle θ is gives the coupling between the two modes.

Notation 2.5.4. A beamsplitter is completely determined by these tree angles. We will often draw it simply as

$$B(\theta, \phi_T, \phi_R) = \left[\begin{array}{cc} |i\rangle & \xrightarrow{\;\theta, \phi_T, \phi_R\;} \langle i| \\ |j\rangle & \longrightarrow \langle j| \end{array}\right] \tag{2.20}$$

This construction of a beamsplitter is by design unitary, which means it preserves energy, and hence no losses occur.[7] We give a short graph-based proof in A.1.

Chapter 3

Boson Sampling

The Boson Sampling problem was first defined by Aaronson & Arkhipov in [2]. We have n identical photons[1] in m modes. In the canonical case, the number of modes is $m = n^2$. We have a unitary linear optical single-photon interferometer represented by the matrix $\mathcal{U} \in \mathbb{U}(m)$ acting on modes. Being a single-photon interferometer, the photons do not *not* interact with each other. Rather, each photon may only interfere with itself.

In the standard setting, the input state is initialised with the first n modes occupied by exactly one photon each, and the rest is unoccupied. Following the set-up in section 2.5, we give the number state $|n\rangle$, as well as qudit representation $\left|\Psi_{\underline{n}}\right\rangle$, below:

$$|\underline{n}\rangle = \prod_{i=1}^{n} \hat{a}_i^\dagger \, |0\rangle^{\otimes m} = |1\rangle^{\otimes n} \otimes |0\rangle^{\otimes (m-n)} \tag{3.1}$$

$$|\underline{n}\rangle \simeq \frac{1}{\sqrt{n! \prod_{i=1}^{m} n_i!}} \sum_{\sigma \in S_n} \sigma \left|\Psi_{\underline{n}}\right\rangle \qquad\qquad \left|\Psi_{\underline{n}}\right\rangle = \bigotimes_{i=1}^{n} |i\rangle \tag{3.2}$$

In (3.2), we use the state $\left|\Psi_{\underline{n}}\right\rangle$ with photons ordered in non-decreasing modes as a convenient representation[11], but recall that the actual physical state must be symmetric. From section 2.5, we also recall the action of the interferometer $\mathcal{U}$:

$$\text{modes:} \quad \hat{a}_i^\dagger \overset{\mathcal{U}}{\longmapsto} \sum_{j=1}^{m} \mathcal{U}_{i,j} \hat{a}_j^\dagger \qquad\qquad \text{qudits:} \quad \left|\Psi_{\underline{n}}\right\rangle \overset{\mathcal{U}}{\longmapsto} \mathcal{U}^{\otimes n} \left|\Psi_{\underline{n}}\right\rangle \tag{3.3}$$

3.1 Derivation of Outcome Probabilities

Our derivation follows that in [20], filling in the missing steps. We denote $|\underline{n}'\rangle$ the output state, the result of sending $|\underline{n}\rangle$ into the interferometer. Remember the interferometer is unitary, so the total number of photons does not change: $n = \sum_i n_i = \sum_i n_i' = n'$. The

[1] in general, we would only need these to be bosons; however, working with photons is practical, and this is the way it was defined

probability of measuring the output state $|\underline{n}'\rangle$ is by the Born rule[5] $\mathbb{P}[\underline{n}'|\underline{n}] = |\langle\underline{n}'|\mathcal{U}|\underline{n}\rangle|^2$. Using (2.15) and (3.3), we express it in qudit states:

$$\mathbb{P}\left[\underline{n}'\middle|\underline{n}\right] = \left|\left(\frac{1}{\sqrt{n!\prod_{i=1}^m n_i'!}}\sum_{\sigma\in S_n}\left\langle\Psi_{\underline{n}'}\right|\sigma^\dagger\right)\mathcal{U}^{\otimes n}\left(\frac{1}{\sqrt{n!\prod_{i=1}^m n_i!}}\sum_{\tau\in S_n}\tau\left|\Psi_{\underline{n}}\right\rangle\right)\right|^2 \tag{3.4}$$

$$= \left|\frac{1}{n!\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\sigma,\tau\in S_n}\left\langle\Psi_{\underline{n}'}\right|\sigma^\dagger\mathcal{U}^{\otimes n}\tau\left|\Psi_{\underline{n}}\right\rangle\right|^2 \tag{3.5}$$

$$= \left|\frac{1}{n!\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\sigma,\tau\in S_n}\left(\bigotimes_{j=1}^m\langle j|^{\otimes n_j'}\right)\sigma^\dagger\mathcal{U}^{\otimes n}\tau\left(\bigotimes_{k=1}^m|k\rangle^{\otimes n_k}\right)\right|^2 \tag{3.6}$$

The tensor products in (3.6) are indexed by modes, with potentially multiple photons in any given mode. The permutations will relabel individual photons, so we need to access those. We will denote $|\psi_i\rangle$ and $|\psi_i'\rangle$ the i^{th} photon of states $\left|\Psi_{\underline{n}}\right\rangle$ and $\left|\Psi_{\underline{n}'}\right\rangle$, respectively.

$$\mathbb{P}\left[\underline{n}'\middle|\underline{n}\right] = \left|\frac{1}{n!\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\sigma,\tau\in S_n}\left(\bigotimes_{j=1}^n\langle\psi_j'|\right)\sigma^\dagger\mathcal{U}^{\otimes n}\tau\left(\bigotimes_{k=1}^n|\psi_k\rangle\right)\right|^2 \tag{3.7}$$

$$= \left|\frac{1}{n!\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\sigma,\tau\in S_n}\left(\bigotimes_{j=1}^n\langle\psi_j'|\right)\mathcal{U}^{\otimes n}\underbrace{\sigma^\dagger\tau}_{\zeta}\left(\bigotimes_{k=1}^n|\psi_k\rangle\right)\right|^2 \tag{3.8}$$

The permutation $\sigma^\dagger$ in (3.7) relabels photons after we have applied $\mathcal{U}$ to each of them. This is the same as first reordering and then applying $\mathcal{U}$. In (3.8), $\sigma^\dagger$ and τ range over S_n, as does their composite ζ. However, there are $|S_n| = n!$ distinct pairs of $\sigma^\dagger$ and τ that compose to each $\zeta\in S_n$, so by removing one summation, we get a factor $n!$.

$$\mathbb{P}\left[\underline{n}'\middle|\underline{n}\right] = \left|\frac{1}{n!\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\,n!\sum_{\zeta\in S_n}\left(\bigotimes_{j=1}^n\langle\psi_j'|\right)\mathcal{U}^{\otimes n}\,\zeta\left(\bigotimes_{k=1}^n|\psi_k\rangle\right)\right|^2 \tag{3.9}$$

$$= \left|\frac{1}{\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\zeta\in S_n}\left(\bigotimes_{j=1}^n\langle\psi_j'|\right)\mathcal{U}^{\otimes n}\left(\bigotimes_{k=1}^n\left|\psi_{\zeta^{-1}(k)}\right\rangle\right)\right|^2 \tag{3.10}$$

In (3.9), photons in the product $\bigotimes_k|\psi_k\rangle$ are reordered by ζ. A photon $|\psi_k\rangle$ that is at position k in the product is moved to index $\zeta(k)$. Then in the reordered state, the photon that is at position k was also put there by ζ, and to find where it was originally, we invert ζ. This way we can directly reorder the product and get (3.10). Then we simplify the inner products (in fact, matrix elements) of tensor products, to get just the product of the matrix elements of corresponding qudits:

$$\mathbb{P}\left[\underline{n}'\middle|\underline{n}\right] = \left|\frac{1}{\sqrt{\prod_{i=1}^m n_i!\,n_i'!}}\sum_{\pi\in S_n}\prod_{j=1}^n\left\langle\psi_j'\right|\mathcal{U}\left|\psi_{\pi(j)}\right\rangle\right|^2 \tag{3.11}$$

In (3.11), we define $\pi := \zeta^{-1}$. We should sum over π^{-1}, but addition commutes and there are no other occurences of π, so we reorder the summation into a more convenient

form. Now, the expression is exactly a permanent from definition 2.4.1. To make this clear, let us define a matrix $\mathcal{V} \in \mathbb{C}^{n\times n}$ by giving its matrix elements:

$$\mathcal{V}_{i,j} \coloneqq \langle \psi_i' | \, \mathcal{U} \, | \psi_j \rangle \qquad\qquad i,j = 1,\dots,n \tag{3.12}$$

Using this matrix, we rewrite (3.11) to

$$\mathbb{P}\left[\underline{n}' \middle| \underline{n}\right] = \left| \frac{1}{\sqrt{\prod_{i=1}^m n_i!\, n_i'!}} \sum_{\pi\in S_n} \prod_{j=1}^n \mathcal{V}_{j,\pi(j)} \right|^2 = \frac{\left|\operatorname{per}\mathcal{V}\right|^2}{\prod_{i=1}^m n_i! \cdot n_i'!} \tag{3.13}$$

and finally, we fill in the initial state defined in (3.1) as $|\underline{n}\rangle = |1\rangle^{\otimes n} \otimes |0\rangle^{\otimes(m-n)}$. Every n_i is either 0 or 1, so all factorials $n_i! = 1$. As $|\underline{n}\rangle$ is fixed, the probability distribution $\mathbb{P}[\underline{n}'|\underline{n}]$ becomes just[20]:

$$\mathbb{P}\left[\underline{n}'\right] = \frac{\left|\operatorname{per}\mathcal{V}\right|^2}{\prod_{i=1}^m n_i'!} \tag{3.14}$$

3.2 Discussion

In (3.14), we see that, up to normalisation, the probability of outcome $|\underline{n}'\rangle$ depends on the permanent of some matrix $\mathcal{V}$. But why should it be the permanent, and what is this matrix?

We start with $\mathcal{V}$. This is a matrix that represents the possible paths that photons can take through the interferometer. In the derivation, we simply defined it from the expression that was already there, but there is meaning in it. Recall from (3.12):

$$\mathcal{V}_{i,j} \coloneqq \langle \psi_i' | \, \mathcal{U} \, | \psi_j \rangle \qquad\qquad i,j = 1,\dots,n$$

where $\left|\psi_j\right\rangle$ is the j^{th} photon of $\left|\Psi_{\underline{n}}\right\rangle = |1\rangle^{\otimes n_1} \otimes |2\rangle^{\otimes n_2} \otimes \cdots \otimes |m\rangle^{\otimes n_m}$, and likewise $|\psi_i'\rangle$ is the i^{th} photon of $\left|\Psi_{\underline{n}'}\right\rangle$. Clearly, the photon modes may repeat (i.e. multiple photons may be in the same mode). Example 3.2.1 shows how this leads to the copying of rows (or columns, but that will never happen with our standard definition of input $|\underline{n}\rangle$).

Example 3.2.1. For $n = n' = 3$, $m = 5$, let

$$\begin{aligned} |\underline{n}\rangle &= |1,1,1,0,0\rangle\,, & \left|\underline{n}'\right\rangle &= |2,0,0,1,0\rangle\,, \\ \left|\Psi_{\underline{n}}\right\rangle &= |1\rangle\,|2\rangle\,|3\rangle\,, & \left|\Psi_{\underline{n}'}\right\rangle &= |1\rangle\,|1\rangle\,|4\rangle\,. \end{aligned}$$

Then, for example, we have these matrix elements: $\mathcal{V}_{1,1} = \langle\psi_1'|\,\mathcal{U}|\psi_1\rangle = \langle 1|\,\mathcal{U}|1\rangle = \mathcal{U}_{1,1}$ and $\mathcal{V}_{2,3} = \langle\psi_2'|\,\mathcal{U}|\psi_3\rangle = \langle 1|\,\mathcal{U}|3\rangle = \mathcal{U}_{1,3}$. The full matrix is:

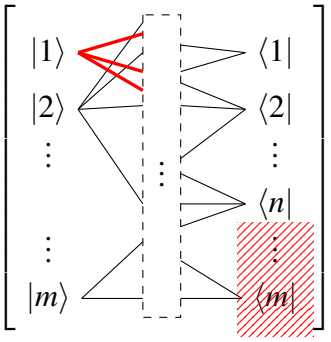
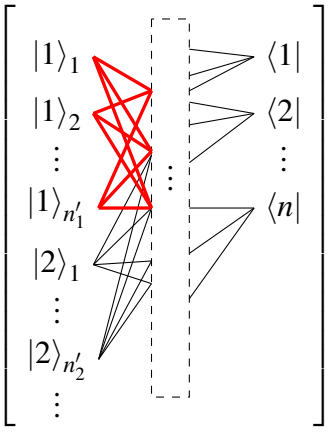
$$\mathcal{V} = \begin{pmatrix} \mathcal{U}_{1,1} & \mathcal{U}_{1,2} & \mathcal{U}_{1,3} \\ \mathcal{U}_{1,1} & \mathcal{U}_{1,2} & \mathcal{U}_{1,3} \\ \mathcal{U}_{4,1} & \mathcal{U}_{4,2} & \mathcal{U}_{4,3} \end{pmatrix}$$

Notice how rows 1 and 2 are the same! This is because the output photons 1 and 2 are in the same mode.

In general, to obtain $\mathcal{V}$, we take n'_i copies of row i in $\mathcal{U}$ to create an $n \times m$ matrix, and from this we take n_j copies of column j. Since $|\underline{n}\rangle = |1^n, 0^{m-n}\rangle$, this operation simply takes the first n columns of $\mathcal{U}$.[20] This suggests that while $\mathcal{U} \in \mathbb{C}^{m\times m}$ is an operator acting on modes (of a single photon), $\mathcal{V} \in \mathbb{C}^{n\times n}$ describes the possible behaviour of each of the n photons. We could say that $\mathcal{V}$ is a realisation of $\mathcal{U}$ on the given input and output occupations. We find the meaning of $\mathcal{V}$ easier to understand on graphs, so we give the following (original) graphical formalization of the construction of $\mathcal{V}$.

Definition 3.2.2 (input and output preparation)**.** Let $\mathcal{U} \in \mathbb{C}^{m\times m}$ be an interferometer matrix and $G(\mathcal{U})$ its graph. We have n photons, input state $|\underline{n}\rangle = |1^n, 0^{m-n}\rangle$, and output $|\underline{n}'\rangle = |n'_1, \ldots, n'_m\rangle$, with $n' = n$. Define the following:

- The *input preparation* function Γ_n^{in} that deletes all input (right-hand-side) vertices $\langle j|$ with $j > n$ in $G(\mathcal{U})$; see the red hatched rectangle in (3.15). This is the deletion of columns.
- The *output preparation* function $\Gamma_{\underline{n}'}^{\text{out}}$ that replaces each output (left-hand-side) vertex $|i\rangle$ with n'_i copies, also copying edges; this is shown with bold red edges for $|1\rangle$ in (3.15). This is the copying of rows.
- The full *preparation* is defined as $\Gamma_{\underline{n}'} = \Gamma_{\underline{n}'}^{\text{out}} \circ \Gamma_n^{\text{in}}$. This gives $\mathcal{V}$.

$G(\mathcal{U}) =$  $\xmapsto{\Gamma_{\underline{n}'}}$  (3.15)

This construction formalizes the idea that the matrix $\mathcal{V} = \Gamma_{\underline{n}'}(\mathcal{U})$ describes the *paths* the photons may take. First, notice that the input vertices (right-hand-side) are precisely the factors of $\left|\Psi_{\underline{n}}\right\rangle = |1\rangle \otimes |2\rangle \otimes \cdots \otimes |n\rangle$, and likewise for the outputs (left-hand-side) and $\left|\Psi_{\underline{n}'}\right\rangle = |\psi'_1\rangle \otimes |\psi'_2\rangle \otimes \cdots \otimes |\psi'_n\rangle$. We have an edge $|\psi'_i\rangle \multimap \langle\psi_j|$ if the photon at position j in mode ψ_j is allowed to reach position i (which has mode ψ_i), and the weight of this edge is the probability amplitude.

These are the possibilities. In the end, every photon starts and ends in a single position of the product, i.e. a single input and a single output vertex. It must pass exactly one of the allowed edges from right to left. This is a perfect matching on the graph! Its probability amplitude is the product of amplitudes of its edges, and we sum over all allowed matchings for any particular $|\underline{n}\rangle, |\underline{n}'\rangle$. We have now arrived at the permanent of $\mathcal{V}$ (cf. sec. 2.4.1). Finally, we will note that this understanding of why a permanent appears agrees exactly with [26, Section 4].

3.3 Classical Simulation

In this work, we adapt the results by Clifford & Clifford from [11]. They give two algorithms to sample the outcomes of a Boson Sampling experiment, which we will label CC-A (for which we give pseudocode) and CC-B. Algorithm CC-A requires time $O(mn3^n)$, while CC-B takes $O(n2^n + \text{poly}(m,n))$ time, where $\text{poly}(m,n) = O(mn^2)$. Both algorithms require $O(m)$ space on top of input storage. Notably, algorithm CC-B creates one sample in time approximately equivalent to that of computing two permanents[11] (see sec. 2.4.2); and it is currently (to our knowledge) the best algorithm to do this for arbitrary inputs.[2] We will not prove these complexities, but in section 5.3.3 we derive the complexity of our algorithm that is based on CC-A.

Both algorithms use the same underlying idea: to create a progressively larger sample by computing marginal probability distributions one photon at a time. We can imagine it as first asking the first photon to which mode it went, then conditioned on that, we ask the second photon, and so on. Concretely, for photon at position $k = 2,\dots,n$, we know the modes of the previous $k-1$ photons. This way, we generate a sequence of photon modes (qudits) $|\psi_1\rangle \cdots |\psi_n\rangle$. In the end, we sort these to get $\left|\Psi_{\underline{n}'}\right\rangle$ as defined in (2.15), which is a convenient representation of $|\underline{n}'\rangle$.

Algorithm CC-A: Boson Sampling by Clifford & Clifford[11, Algorithm A]

Input:

- $m,n \in \mathbb{Z}_+$, with $n \le m$
- matrix $\mathcal{U} \in \mathbb{U}(m)$ a Haar random unitary representing an interferometer.

Result: a single sample $\left|\Psi_{\underline{n}'}\right\rangle$

1 **for** $k \leftarrow 1,\dots,n$ **do**
2 **for** $i \in [m]$ **do**
3 $w_k(i) \leftarrow \sum_{\substack{C\subseteq[n]\\|C|=k}} \left|\text{per}\left(\widetilde{\Gamma}^{\text{out}}_{(\psi_1,\dots,\psi_{k-1},i)}\ \widetilde{\Gamma}^{\text{in}}_{C}\ \mathcal{U}\right)\right|^2$
4 sample $\psi_k \leftarrow i$ using an (unnormalised) pmf $w_k(i)$
5 $\left|\Psi_{\underline{n}'}\right\rangle \leftarrow$ sort $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$ in non-decreasing order
6 **return** $\left|\Psi_{\underline{n}'}\right\rangle$

Algorithm CC-A works by computing an (unnormalised) probability mass function (pmf) $w_k : [m] \to \mathbb{R}$ for each photon k in the sequence, where $k \in [n]$. This is a probability that photon k is in mode i: $|\psi_k\rangle = |i\rangle$; and is conditioned on the already known modes of all previous photons. This function w_k is used at each stage to sample the mode of photon k.

We will follow the argument in [11], filling in our own steps and commenting on the physical intuition. From (3.14), we have the probability of a particular sample $|\underline{n}'\rangle$:

$$\mathbb{P}\left[\underline{n}'\right] = \frac{\left|\text{per}\,\mathcal{V}\right|^2}{\prod_{i=1}^m n_i'!} = \frac{\left|\text{per}\,\Gamma_{\underline{n}'}(\mathcal{U})\right|^2}{\prod_{i=1}^m n_i'!} = \frac{\left|\text{per}\,\Gamma^{\text{out}}_{\underline{n}'}\,\Gamma^{\text{in}}_{n}(\mathcal{U})\right|^2}{\prod_{i=1}^m n_i'!} \tag{3.16}$$

[2] Clifford and Clifford published another algorithm that runs much faster if $m \propto n$. In particular, if $m = n$, then the average time is $O(n \cdot 1.69^n)$.[12]

where we have fully expanded the definition of input and output preparation $\Gamma_{\underline{n}}$. The number state corresponds to $\left|\Psi_{\underline{n}'}\right\rangle = |\psi_1\rangle \cdots |\psi_n\rangle$ which is a non-decreasing sequence of photon modes. To simplify notation, we will denote this sequence as $\underline{z} \in [m]^n$. We rewrite the probability in this representation:

$$q(\underline{z}) \coloneqq \mathbb{P}[\underline{z}] = \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{z}} \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{\prod_{i=1}^m n_i'!} = \mathbb{P}\left[\underline{n}'\right] \tag{3.17}$$

which is now in the qudit representation. These are the same $\underline{z}$ and $q(\underline{z})$ as used in [11]. We have also redefined the preparation functions Γ for this representation:

Definition 3.3.1 (input preparation (qudits))**.** Let $\mathcal{U} \in \mathbb{C}^{\mathcal{R}\times\mathcal{C}}$ (with $\mathcal{R} = \mathcal{C} = [m]$) be an interferometer matrix, $\mathcal{C}' \subseteq \mathcal{C}$ be a subset of input modes (columns), and $\underline{r} \in \mathcal{R}^{|\mathcal{C}'|}$ a sequence of output mode (rows) of length $|\mathcal{C}'|$. The input preparation $\widetilde{\Gamma}^{\text{in}}_{\mathcal{C}'}$ deletes all input modes that are not in $\mathcal{C}'$, and the output preparation $\widetilde{\Gamma}^{\text{out}}_{\underline{r}}$ takes a matrix $\mathcal{U}$ and creates a new one with rows $\mathcal{U}_{r_1}, \mathcal{U}_{r_2}, \ldots, \mathcal{U}_{r_{|\mathcal{C}'|}}$.

Note that we do not require $\underline{r}$ to be sorted in the non-decreasing order: we are now looking at the particular measurements of photon modes, not the number states. The probability of a particular sample $\underline{r} \in [m]^n$ is:[11]

$$p(\underline{r}) \coloneqq \mathbb{P}[\underline{r}] = \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{r}} \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{n!} = \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{z}} \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{n!} = p(\underline{z}) \tag{3.18}$$

where we used the fact that permanents are invariant under row or column permutation (lemma 2.4.5) to equate this to probability of getting the sorted $\underline{z}$ corresponding to $\underline{r}$ (we write this $\underline{z} \simeq \underline{r}$). Still dealing with a full sample of n photons, the input preparation is over the full $[n]$ (this will change later). For any number state $|\underline{n}'\rangle$ represented by $\underline{z}$ (in qudits), we have $\binom{n}{n_1', \ldots, n_m'}$ different possible $\underline{r}$, i.e. distinct orderings of photons (recall that n_i' is the number of photons in mode i). Clearly, $\underline{r}$ are just different configurations (microstates) of the number state represented by $\underline{z}$, and we show that their probabilities sum to $q(\underline{z})$:

$$\sum_{\underline{r} \simeq \underline{z}} p(\underline{r}) \overset{(3.18)}{=} p(\underline{z}) \sum_{\underline{r} \simeq \underline{z}} 1 = \binom{n}{n_1', n_2', \ldots, n_m'} \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{z}} \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{n!} \tag{3.19}$$

$$= \frac{n!}{\prod_{i=1}^m n_i'!} \cdot \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{z}} \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{n!} \overset{(3.17)}{=} q(\underline{z}) \tag{3.20}$$

Looking at qudit measurements like this allows us to work with partial samples of $k < n$ photons. The definition of $\widetilde{\Gamma}^{\text{in}}_{\mathcal{C}'}$ allows us to select which of the incoming photons were measured in the partial sample. This is important, because if we only measure the first k photons, we cannot assume they are the same ones that were in the first k input modes (recall we have single-photon modes at input). Instead, the probability of measuring some partial sample $\underline{r} \coloneqq (r_1, \ldots, r_k) \in [m]^k$ of k photons will have to sum over the $\binom{n}{k}$ possible subsets of input modes from which these photons came:[3][11]

[3] we drop the prime from $\mathcal{C}'$ here, just to simplify notation

Lemma 3.3.2 (partial sample)**.** *The marginal pmf of a partial sample of $k \in [n]$ photons is*

$$p(r_1,\dots,r_k) = \frac{(n-k)!}{n!} \sum_{\mathcal{C}\subseteq[n];\,|\mathcal{C}|=k} \left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{(r_1,\dots,r_k)}\, \widetilde{\Gamma}^{\text{in}}_{\mathcal{C}}(\mathcal{U})\right|^2 \tag{3.21}$$

Proof. Let $\underline{r} = (r_1,\dots,r_n)$ be the full sample and $\underline{r}' = (r_{k+1},\dots,r_n)$ the part that we do not want. Then

$$p(r_1,\dots,r_k) = \sum_{\underline{r}'} p(\underline{r}) = \sum_{\underline{r}'} \frac{\left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{\underline{r}}\, \widetilde{\Gamma}^{\text{in}}_{[n]}(\mathcal{U})\right|^2}{n!} \tag{3.22}$$

$$= \frac{1}{n!} \sum_{\underline{r}'} \left(\sum_{\sigma\in S_n} \prod_{j=1}^{n} \mathcal{U}_{r_j,\sigma_j} \right) \left(\sum_{\tau\in S_n} \prod_{\ell=1}^{n} \mathcal{U}_{r_\ell,\tau_\ell} \right)^* \tag{3.23}$$

where we have expanded the definition of permanent 2.4.1, used that for $x \in \mathbb{C}$ the magniture squared is $|x|^2 = x\cdot x^*$. Recall that $\widetilde{\Gamma}^{\text{out}}_{\underline{r}}(M)$ gives a matrix with rows $M_{r_1},\dots,M_{r_n}$. For efficiency, we write $\sigma_j \equiv \sigma(j)$, likewise for τ.

$$p(r_1,\dots,r_k) = \frac{1}{n!} \sum_{\underline{r}'} \sum_{\sigma,\tau\in S_n} \prod_{j=1}^{n} \mathcal{U}_{r_j,\sigma_j} \mathcal{U}^*_{r_j,\tau_j} = \frac{1}{n!} \sum_{\underline{r}'} \sum_{\sigma,\tau\in S_n} \prod_{j=1}^{n} \mathcal{U}^\dagger_{\tau_j,r_j} \mathcal{U}_{r_j,\sigma_j} \tag{3.24}$$

$$= \frac{1}{n!} \sum_{\sigma,\tau\in S_n} \left(\prod_{j=1}^{k} \mathcal{U}^\dagger_{\tau_j,r_j} \mathcal{U}_{r_j,\sigma_j} \right) \sum_{\underline{r}'} \prod_{\ell=k+1}^{n} \mathcal{U}^\dagger_{\tau_\ell,r_\ell} \mathcal{U}_{r_\ell,\sigma_\ell} \tag{3.25}$$

We have rearranged the expressions. In the right hand side of (3.24), we used the unitarity of $\mathcal{U}$, giving us $\mathcal{U}^*_{r_j,\tau_j} = \mathcal{U}^\dagger_{\tau_j,r_j}$. In (3.25), we moved the terms not summed over outside of the summation over $\underline{r}' = (r_{k+1},\dots,r_n)$.

$$p(r_1,\dots,r_k) = \frac{1}{n!} \sum_{\sigma,\tau\in S_n} \left(\prod_{j=1}^{k} \mathcal{U}^\dagger_{\tau_j,r_j} \mathcal{U}_{r_j,\sigma_j} \right) \prod_{\ell=k+1}^{n} \sum_{r_\ell=1}^{m} \mathcal{U}^\dagger_{\tau_\ell,r_\ell} \mathcal{U}_{r_\ell,\sigma_\ell} \tag{3.26}$$

$$= \frac{1}{n!} \sum_{\sigma,\tau\in S_n} \left(\prod_{j=1}^{k} \mathcal{U}^\dagger_{\tau_j,r_j} \mathcal{U}_{r_j,\sigma_j} \right) \prod_{\ell=k+1}^{n} [\mathcal{U}^\dagger \mathcal{U}]_{\tau_\ell,\sigma_\ell} \tag{3.27}$$

In (3.26), we exchanged the product and sum by distributivity. Only one r_ℓ appears now, but the product ranges over ℓ. By unitarity $\mathcal{U}^\dagger \mathcal{U} = \mathbb{1}$. This means the only non-zero terms are those where $\sigma_\ell = \tau_\ell$ for all $\ell > k$. We partition the permutations as $\sigma = \mu \oplus \zeta$ and $\tau = \nu \oplus \zeta$ into a shared part $\zeta \in S_{n-k}$ (this is the $\sigma_\ell = \tau_\ell$), and smaller permutations $\mu,\nu \in S_k$. This means we also partition the set $[n]$ on which σ,τ act into a subset $\mathcal{C}$ of size k on which μ,ν act, and $[n] \setminus \mathcal{C}$ acted upon by ζ. We sum over partitions $\mathcal{C}$, and for each there are $(n-k)!$ possible permutations ζ; this gives the new factor in (3.28).

$$p(r_1,\dots,r_k) = \frac{(n-k)!}{n!} \sum_{\mathcal{C}\subseteq[n];\,|\mathcal{C}|=k} \left(\sum_{\mu\in S_{\mathcal{C}}} \prod_{j=1}^{k} \mathcal{U}_{r_j,\mu_j} \right) \left(\sum_{\nu\in S_{\mathcal{C}}} \prod_{\ell=1}^{k} \mathcal{U}_{r_\ell,\nu_\ell} \right)^* \tag{3.28}$$

$$= \frac{(n-k)!}{n!} \sum_{\mathcal{C}\subseteq[n];\,|\mathcal{C}|=k} \left|\operatorname{per} \widetilde{\Gamma}^{\text{out}}_{(r_1,\dots,r_k)}\, \widetilde{\Gamma}^{\text{in}}_{\mathcal{C}}(\mathcal{U})\right|^2 \tag{3.29}$$

In (3.28), we have also split apart the permutations μ,ν, and arrived back at the expression of a permanent.[11] ∎

Lemma 3.3.3. *For $\underline{r} \in [m]^n$ as defined above, the probability is[11]*

$$p(\underline{r}) = p(r_1) \cdot p(r_2|r_1) \cdot p(r_3|r_1, r_2) \cdots p(r_n|r_1, \ldots, r_{n-1}) \tag{3.30}$$

Proof. Here, we use the chain rule for probabilities $\mathbb{P}[A,B] = \mathbb{P}[A|B] \cdot \mathbb{P}[B]$:

$$\begin{aligned} p(\underline{r}) &= p(r_n|r_1, \ldots, r_{n-1}) \cdot p(r_1, \ldots, r_{n-1}) && (3.31)\\ &= p(r_n|r_1, \ldots, r_{n-1}) \cdot p(r_{n-1}|r_1, \ldots, r_{n-2}) \cdot p(r_1, \ldots, r_{n-2}) && (3.32) \end{aligned}$$

and we expand this until we have $p(r_1) \cdot p(r_2|r_1) \cdot p(r_3|r_1, r_2) \cdots p(r_n|r_1, \ldots, r_{n-1})$. ■

Lemma 3.3.3 gives us the ability to generate the sample progressively, one photon at a time, each conditioned on all the previous photon modes. The $p(r_1, \ldots, r_k)$ in (3.21) is exactly the pmf w_k that is computed at line 3 of Algorithm CC-A (but w_k is not normalised). Hence, we have shown the idea of the algorithm that we aim to extend.

3.3.1 Permuting Input Modes

With what we have done so far, we always have to account for the $\binom{n}{k}$ possible subsets of input modes from which the k sampled photons came. This introduces more complexity. A clever trick done in [11] allows us to randomly permute the input modes instead.

We recall equation (3.21) which gives the probability distribution of partial samples of k photons:

$$p(r_1, \ldots, r_k) = \frac{(n-k)!}{n!} \sum_{\mathcal{C} \subseteq [n];\, |\mathcal{C}|=k} \left| \operatorname{per} \widetilde{\Gamma}^{\text{out}}_{(r_1,\ldots,r_k)} \, \widetilde{\Gamma}^{\text{in}}_{\mathcal{C}}(\mathcal{U}) \right|^2$$

Instead of summing over subsets $\mathcal{C}$, we sum over permutations $\alpha \in \mathrm{S}_n$ (not m, only the occupied inputs!), and always take $\mathcal{C} = \alpha([k]) = \{\alpha(c) | c \in [k]\}$. This corresponds to taking the first k columns of a permuted matrix.

$$\begin{aligned} p(r_1, \ldots, r_k) &= \frac{(n-k)!}{n!} \frac{1}{k!(n-k)!} \sum_{\alpha \in \mathrm{S}_n} \left| \operatorname{per} \Gamma^{\text{out}}_{(r_1,\ldots,r_k)} \, \Gamma^{\text{in}}_{\alpha([k])}(\mathcal{U}) \right|^2 && (3.33)\\ &= \sum_{\alpha \in \mathrm{S}_n} \frac{1}{n!} \left[\frac{1}{k!} \left| \operatorname{per} \Gamma^{\text{out}}_{(r_1,\ldots,r_k)} \, \Gamma^{\text{in}}_{\alpha([k])}(\mathcal{U}) \right|^2 \right] && (3.34)\\ &= \mathbb{E}_{\alpha \in \mathrm{S}_n} \underbrace{\left[\frac{1}{k!} \left| \operatorname{per} \Gamma^{\text{out}}_{(r_1,\ldots,r_k)} \, \Gamma^{\text{in}}_{\alpha([k])}(\mathcal{U}) \right|^2 \right]}_{\phi(r_1,\ldots,r_k|\alpha)} && (3.35) \end{aligned}$$

In (3.33), we sum over $\alpha \in \mathrm{S}_n$. For each subset $\mathcal{C}$, there are $k!(n-k)!$ corresponding permutations α, so we need to divide by this factor. After rearranging, in (3.34), we recognize that the summation is an expectation over uniformly distributed permutations $\alpha \in \mathrm{S}_n$ (each with probability $1/n!$) of the function in brackets, which Clifford & Clifford[11] call $\phi(r_1, \ldots, r_k|\alpha)$.

Equation (3.35) allows us to randomly permute the matrix columns at the beginning and then compute the probabilities without needing to sum over subsets of inputs. This is exactly what the algorithm CC-B does.[11, Algorithm B] We will also use this trick in our algorithm.

Chapter 4

Tree Decompositions

We wish to sample from a shallow interferometer. As we will see later, in lemma 5.1.6, this means the matrix of that interferometer is sparse, which means many of its elements are zero. This chapter is about tree decompositions, which are not by themselves related to Boson Sampling. However, these are tools from which we will benefit greatly.

In [9], Cifuentes and Parrilo give an algorithm that can compute the permanent of a sparse matrix very efficiently by taking advantage of the sparsity structure. Essentially, in a sparse matrix, many terms in the permanent vanish, and with proper preparation, we can avoid even looking at those terms, thus reducing the computation time.

4.1 Basics

Definition 4.1.1 ((bipartite) tree decomposition[9])**.** Let $G = (V = (\mathcal{R}, \mathcal{C}), E \subseteq \mathcal{R} \times \mathcal{C})$ be a bipartite graph. A tree decomposition $\mathrm{T} = (T, \rho, \kappa)$ is a triple containing:

- the underlying tree T, which is rooted in $\mathrm{root}(T)$, and for each *node* $t \in T$, we denote $\mathrm{ch}(t)$ the unordered set of its children; and
- the functions $\rho : T \to \mathcal{P}(\mathcal{R})$ and $\kappa : T \to \mathcal{P}(\mathcal{C})$ that label the nodes with subsets of $\mathcal{R}$ and $\mathcal{C}$; we will say that a node $t \in T$ *contains* the elements $\rho(t)$ and $\kappa(t)$.

A valid tree decomposition must satisfy the following axioms:

(T1) The union of labels over the entire tree must cover all vertices of the graph:

$$\bigcup_{t \in T} \rho(t) = \mathcal{R} \qquad\qquad \bigcup_{t \in T} \kappa(t) = \mathcal{C}$$

(T2) For every edge $(r, c) \in E$, there exists a node $t \in T$ that contains both its endpoints:

$$r \in \rho(t) \qquad\qquad c \in \kappa(t)$$

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

For a graph G, we will write $\mathcal{T}(G)$ for the set of all valid tree decompositions of this graph.

Note the tree is written T, while the tree decomposition built upon it is written upright: T. Also note that in the definition in [9], Cifuentes & Parrilo use different symbols: A for $\mathcal{R}$, α for ρ, X for $\mathcal{C}$, and χ for κ. We use these to to represent **r**ows and **c**olumns of a matrix, so we choose a different notation, in the hope that is it easier to remember.[1] For clarity, we make also the distinction in words used: whenever we say *node*, it means the node of a tree decomposition; and when we use *vertex*, it will always be a vertex of the original graph.

We give an example of a tree decomposition from a matrix graph in figure 4.1.

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & 0 \\ 0 & 0 & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{pmatrix} \qquad G(M) = \left[\begin{array}{c|c} |1\rangle & \langle 1| \\ |2\rangle & \langle 2| \\ |3\rangle & \langle 3| \end{array}\right]$$

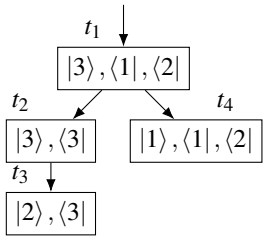


Figure 4.1: Example of a tree decomposition $\mathrm{T} = (T, \rho, \kappa)$ of $G(M)$

Definition 4.1.2 (subtree)**.** If T is a tree rooted at root(t), and $t \in T$ is some node, then we call T_t the subtree of T that is rooted at t and contains all its descendants (with the same structure as in T). We can see T as a directed graph, with arrows oriented from parent $\rightarrow$ child; then T_t is exactly what we get by finding t, keeping t and everything reachable from it, and throwing out everything else. For example, in figure 4.1, the subtree $T_{t_2} = \{t_2, t_3\}$.

Note 4.1.3 (forests)**.** If the graph we are decomposing has multiple connected components that are disconnected from each other, the axioms would only require that we have a forest, where each disjoint tree corresponds to one of those components. We need the decomposition to be a tree, so we must connect the disjoint trees of the forest; any way will do, and everything will still work. This corresponds to having block-like structures in the matrix (up to maybe permuting rows and columns).

A tree decomposition captures the connectedness (or, even more importantly, the disconnectedness) of a graph. In particular, the axiom (T2) ensures that all edges are contained, and this is central. It does not exclude nonexistent edges from being hinted at, i.e. by putting vertices that do not have an edge into the same tree node. Then a valid tree decomposition of any graph $G = ((\mathcal{R}, \mathcal{C}), E)$ is the following trivial one: $\mathrm{T} = (T = \{t\}, \rho : t \mapsto \mathcal{R}, \kappa : t \mapsto \mathcal{C})$. Here, T has a tree of single node, and all vertices are contained in that node. This clearly satisfies axioms, but it is not a good tree decomposition.

[1] Cifuentes and Parrilo use it for the same purpose; it may be that A, X for rows and columns is a convention unknown to us.

4.2 Treewidth

What is a good tree decomposition? It is one that captures the structure of the graph correctly, but does not contain more vertices than necessary in each node. In particular, computing with trees is easier than computing with graphs containing cycles (in fact, this is sometimes NP-hard).[25] We want a tree decomposition to capture the essential *treeness* of a graph, and we quantify what that means using the following notion:

Definition 4.2.1 (treewidth[9, 25])**.** Let G be a bipartite graph and $\mathrm{T} = (T, \rho, \kappa)$ some tree decomposition of G. Then the treewidth of the tree decomposition is

$$\mathrm{tw}(\mathrm{T}) \coloneqq \max_{t \in T} \left\{ |\rho(t)| + |\kappa(t)| \right\} - 1, \tag{4.1}$$

that is the size of the *largest* node t (in terms of its contents $\rho(t)$ and $\kappa(t)$) minus one. Conversely, the treewidth of the graph G is the *minimum* treewidth possible with any tree decomposition:

$$\widehat{\mathrm{tw}}(G) \coloneqq \min_{\mathrm{T} \in \mathcal{T}(G)} \mathrm{tw}(\mathrm{T}), \tag{4.2}$$

which we denote with a hat to distinguish it from the treewidth of a particular decomposition.

The minus sign in (4.1) was chosen arbitrarily, so that the treewidth of a tree T is $\widehat{\mathrm{tw}}(T) = 1$. This is because treewidth measures how different a graph is from a tree. The treewidth of the decomposition in figure 4.1 is $\mathrm{tw}(\mathrm{T}) = 2$: the graph $G(M)$ is almost a tree, but there is a cycle, namely $|1\rangle - \langle 1| - |3\rangle - \langle 2| - |1\rangle$.

There may be many valid tree decompositions of any graph. In particular, equation (4.2) suggests that finding a good one is a minimization task in its own right. Tree decompositions, and especially the bounds of treewidth, have been explored for various classes of graphs. However, in particular for bipartite graphs, we do not have a good way to bound the treewidth or compute a low-treewidth decomposition.[25] Recall that our graph representation of matrices is bipartite! This may pose a problem, and we need more structure in our matrices.

Cifuentes & Parrilo give a few examples of low-treewidth tree decompositions, and among them is a tree decomposition of a graph of a banded matrix.[9] Bandedness turns out to be exactly what we need, and this decomposition will be very useful for our algorithm. We describe it in section 5.1.2. Additionally, we have explored two other methods for generating tree decompositions: one is a greedy algorithm, and one performs *decycling* of the line graph of $G(M)$. We have not used them, and so they are only shortly described in A.2, A.3.

4.3 Permanent From a Tree Decomposition

While (T1) and (T2) capture all vertices and edges, it is together with (T3) that we are able to extract the structure of a graph. At a high level, (T3) causes the edges that share a vertex in the graph to be also connected by a path in the tree decomposition, and this will be useful for the permanent calculation.

Algorithm CP-2: BipartitePermanent

Input:
- a matrix $M \in \mathbb{C}^{n\times n}$, or equivalently its bipartite graph $G = G(M)$,
- a tree decomposition $\mathrm{T} = (T, \rho, \kappa)$ of G

Result: the permanent $\operatorname{per} M$

$order \leftarrow$ list of nodes in T, in topological order, starting from leaves
for $t \in order$ **do**
  $Q[t], P[t] \leftarrow$ new tables
  $Q[t](R,C) \leftarrow \operatorname{per}\left(M\big|_{R,C}\right)$ for $R \subseteq \rho(t)$ and $C \subseteq \kappa(t)$
  if *t is a leaf, i.e.* $\operatorname{ch}(t) = \varnothing$ **then**
    $P[t] \leftarrow Q[t]$
  else
    EvalResursion(t)
return $P[r](\rho(r), \kappa(r)) = \operatorname{per} M$, where $r = \operatorname{root}(T)$

Procedure EvalResursion(t)

$ch \leftarrow |\operatorname{ch}(t)|$
for $c_j \in \operatorname{ch}(t)$ **do**
  $\begin{cases} \Delta_j^\rho & \leftarrow \rho(c_j) \setminus \rho(t) \\ I_j^\rho & \leftarrow \rho(c_j) \cap \rho(t) \end{cases} \quad \begin{cases} \Delta_j^\kappa & \leftarrow \kappa(c_j) \setminus \kappa(t) \\ I_j^\kappa & \leftarrow \kappa(c_j) \cap \kappa(t) \end{cases}$
  $Q'[t\,|\,c_j], Q''[t \to c_j] \leftarrow$ new tables
  for $R \subseteq I_j^\rho$ *and* $C \subseteq I_j^\kappa$ **do**
    $Q'[t\,|\,c_j](R,C) \leftarrow (-1)^{|R|} \cdot Q[t](R,C)$
    $Q''[t \to c_j](R,C) \leftarrow P[c_j](R \cup \Delta_j^\rho, C \cup \Delta_j^\kappa)$
for $R \subseteq \rho(t)$ *and* $C \subseteq \kappa(t)$ **do**
  $P[t](R,C) \leftarrow$ Subset Convolution$(R, C; Q[t], Q'[t|c_1], Q''[t \to c_1], \ldots, Q'[t|c_{ch}], Q''[t \to c_{ch}])$

Cifuentes & Parrilo give in [9, Algorithm 2] a dynamic programming algorithm that can compute the permanent of an $n \times n$ matrix by using the tree decomposition of its graph in time $\widetilde{O}(n2^\omega)$, where ω is the treewidth of the decomposition. The notation $\widetilde{O}$ is like the asymptotic complexity O, but it also hides polynomial factors of ω.

We give the pseudocode in Algorithm CP-2 and procedure EvalResursion. However, we do not provide the proof of this algorithm nor its running time: it is very complex and relies on other previous work that is out of scope for this project (in particular, the paper [6] on subset convolutions). Instead, we demonstrate the algorithm with an example; we believe this will convey the intuition more efficiently.

The algorithm builds the permanent of the whole matrix from permanents of submatrices.[9] The tree decomposition gives us the ability to avoid the zeros in the matrix, so we will not even need to look at terms that would vanish.

We will use the tree decomposition in figure 4.1 as our example. Before delving in, we recall the matrix $M \in \mathbb{C}^{\mathcal{R}\times\mathcal{C}}$ (with $\mathcal{R} = \mathcal{C} = [3]$), and we assume that $M_{i,j}$ are nonzero elements, unless we explicitly put a zero into the matrix below. We give also its permanent to know what we expect to compute. In figure 4.2, we recall its tree decomposition $\mathrm{T} = (T, \rho, \kappa)$, and annotate it with data computed in the following subsections.

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & 0 \\ 0 & 0 & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{pmatrix} \qquad \operatorname{per} M = M_{1,2}M_{2,3}M_{3,1} + M_{1,1}M_{2,3}M_{3,2}. \tag{4.3}$$

In the following subsections, we compute multiple dynamic programming tables. The main ones are Q, relating a single node to permanents of matrices it contains, and P, relating to a subtree in a similar way. We find the final result in the P table of the root, in the part that represents the submatrix of all rows and columns contained in the root.

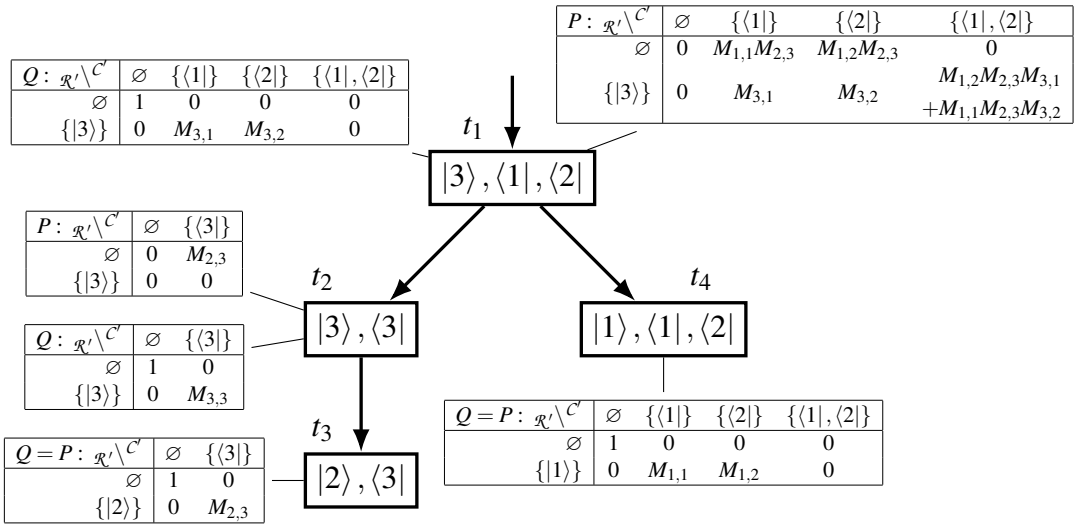


Figure 4.2: Tree decomposition from figure 4.1, annotated with the dynamic programming tables from the execution of Algorithm CP-2.

4.3.1 Simple Steps

Definition 4.3.1 (local table[9])**.** For a node of the tree $t \in T$, we define the table as

$$\begin{aligned} Q[t] : \mathcal{P}\big(\rho(t)\big) \times \mathcal{P}\big(\kappa(t)\big) &\to \mathbb{C} \\ (\mathcal{R}', \mathcal{C}') &\mapsto \operatorname{per}\Big(M\big|_{\mathcal{R}',\mathcal{C}'}\Big) \end{aligned} \tag{4.4}$$

This table assigns to all subsets $\mathcal{R}' \subseteq \rho(t)$ of rows and $\mathcal{C}' \subseteq \kappa(t)$ of columns contained in t (in other works, the submatrices $M\big|_{\mathcal{R}',\mathcal{C}'}$) the values of their permanents. This way, the tables Q are *local*, each $Q[t]$ only deals with the node t and nothing else.

Recall from lemma 2.4.3 that if $\mathcal{R}' = \mathcal{C}' = \varnothing$, then $\operatorname{per} M\big|_{\mathcal{R}',\mathcal{C}'} = 1$; and if the submatrix is not square ($|\mathcal{R}'| \neq |\mathcal{C}'|$), then $\operatorname{per} M\big|_{\mathcal{R}',\mathcal{C}'} = 0$. In figure 4.2, these tables are labeled just Q, and are connected to their corresponding node.

4.3.2 Recursive Steps

Having computed Q, we now need the recursion to happen. We go up the tree, in a topological ordering starting from the leaves. For each node $t \in T$, we will compute another table called $P[t] : \mathcal{P}\big(\rho(t)\big) \times \mathcal{P}\big(\kappa(t)\big) \to \mathbb{C}$. These have the same type as $Q[t]$, assigning numbers to subsets of rows and columns contained in t. Their difference from Q is that while $Q[t]$ has information only about the submatrices it contains, $P[t]$ knows about the subtree T_t. Clearly, since a leaf $\ell \in T$ no children, the tables are the same: $Q[\ell] = P[\ell]$.[9]

4.3.2.1 Definitions of Recursion Tables

If we are computing the recursive step at a node t (not leaf), we compute the intersections and differences in the content of t compared to its children $c_1, \ldots, c_k \in \mathrm{ch}(t)$:

Definition 4.3.2 (intersections and differences[9])**.** For every child c_j of t, we define:

$$\Delta^{\rho}_{c_j} \coloneqq \rho(t_3) \setminus \rho(t_2) \qquad \Lambda^{\rho}_{c_j} \coloneqq \rho(t_3) \cap \rho(t_2) \tag{4.5}$$

$$\Delta^{\kappa}_{c_j} \coloneqq \kappa(t_3) \setminus \kappa(t_2) \qquad \Lambda^{\kappa}_{c_j} \coloneqq \kappa(t_3) \cap \kappa(t_2) \tag{4.6}$$

Definition 4.3.3 (auxiliary tables[9])**.** For every child c_j of t, we define the table $Q'[t|c_j]$, where we use (our own) notation to suggest that this is a table for node t *given that* we are looking at its child c_j. It is defined as:

$$\begin{aligned} Q'[t|c_j] : \mathcal{P}(\Lambda^{\rho}_{c_j}) \times \mathcal{P}(\Lambda^{\kappa}_{c_j}) &\to \mathbb{C} \\ (\mathcal{R}', \mathcal{C}') &\mapsto (-1)^{|\mathcal{R}'|} \cdot Q[t](\mathcal{R}', \mathcal{C}') \end{aligned} \tag{4.7}$$

We define also $Q''[t \to c_j]$, which corresponds to looking into the child c_j from t:

$$\begin{aligned} Q''[t \to c_j] : \mathcal{P}(\Lambda^{\rho}_{c_j}) \times \mathcal{P}(\Lambda^{\kappa}_{c_j}) &\to \mathbb{C} \\ (\mathcal{R}', \mathcal{C}') &\mapsto P[c_j]\left(\mathcal{R}' \cup \Delta^{\rho}_{c_j}, \mathcal{C}' \cup \Delta^{\kappa}_{c_j}\right) \end{aligned} \tag{4.8}$$

Definition 4.3.4 (permanent table)**.** For a node t, the permanent table $P[t]$ joins together the auxiliary tables of all its children $c_1, \ldots, c_k \in \mathrm{ch}(t)$ by subset convolution. It has the same type as $Q[t]$, that is $P[t] : \mathcal{P}(\rho(t)) \times \mathcal{P}(\kappa(t)) \to \mathbb{C}$.

$$P[t](\mathcal{R}', \mathcal{C}') \coloneqq \sum_{\substack{\mathcal{R}_0, \ldots, \mathcal{R}_{2k};\ \mathcal{C}_0, \ldots, \mathcal{C}_{2k} \\ (\mathcal{R}' = \mathcal{R}_0 \sqcup \cdots \sqcup \mathcal{R}_2) \\ (\mathcal{C}' = \mathcal{C}_0 \sqcup \cdots \sqcup \mathcal{C}_2)}} Q[t_2] \cdot Q'[t_2|c_1] \cdot Q''[t_2 \to c_1] \cdots Q'[t_2|c_k] \cdot Q''[t_2 \to c_k] \tag{4.9}$$

We omit arguments for clarity, but each factor gets a pair of $\mathcal{R}_i, \mathcal{C}_i$.

4.3.2.2 Recursive Step $t_3 \to t_2$

The first recursive step we compute is (by virtue of topological ordering) the table $P[t_2]$. It is a simple one: looking at the figure 4.2, we see that the node t_2 has only one child, $\mathrm{ch}(t_2) = \{t_3\}$. The differences are $\Delta^{\rho}_{t_3} = \{|2\rangle\}$, $\Delta^{\kappa}_{c_j} = \varnothing$, and the intersections are $\Lambda^{\rho}_{t_3} = \varnothing$, $\Lambda^{\kappa}_{t_3} = \{\langle 3|\}$. Then we compute the auxiliary tables:

$Q'[t_2\|t_3] : {}_{\mathcal{R}'}\backslash^{\mathcal{C}'}$	$\varnothing$	$\langle 3\|$	$Q''[t_2 \to t_3] : {}_{\mathcal{R}'}\backslash^{\mathcal{C}'}$	$\varnothing$	$\langle 3\|$
$\varnothing$	1	0	$\varnothing$	0	$M_{2,3}$

Both assign values to submatrices of rows and columns shared between t_2 and t_3. The table $Q'[t_2|t_3]$ reuses values from $Q[t_2]$, but makes some of them (exactly when $|\mathcal{R}'|$ is odd) negative. This is a preparation for the subset convolution. The table $Q''[t_2 \to t_3]$ pulls values up the tree from $P[t_3]$. Notice, however, that computing $Q''[t_2 \to t_3](\mathcal{R}', \mathcal{C}')$ does not just take $P[t_3](\mathcal{R}', \mathcal{C}')$; it extends the argument by the difference sets $\Delta^{\rho,\kappa}$. This way, we are reaching possibly a different part of matrix M.

Lastly, we compute the table $P[t_2]$ by subset convolution of $Q[t_2], Q'[t_2|t_3]$ and $Q''[t_2 \to t_3]$. For $P[t_2](\mathcal{R}' \subseteq \rho(t), \mathcal{C}' \subseteq \kappa(t))$, we sum over the partitions of these $\mathcal{R}', \mathcal{C}'$, such that each of the tables gets one subset. For example, we solve it for $\mathcal{R}' = \varnothing$ and $\mathcal{C}' = \{\langle 3|\}$:

$$P[t_2](\varnothing, \{\langle 3|\}) = \sum_{\substack{\mathcal{C}_0,\mathcal{C}_1,\mathcal{C}_2 \\ (\{\langle 3|\} = \mathcal{C}_0 \sqcup \mathcal{C}_1 \sqcup \mathcal{C}_2)}} Q[t_2](\varnothing, \mathcal{C}_0) \cdot Q'[t_2|t_3](\varnothing, \mathcal{C}_1) \cdot Q''[t_2 \to t_3](\varnothing, \mathcal{C}_2) \tag{4.10}$$

$$= \underbrace{Q[t_2](\varnothing, \{\langle 3|\})}_{0} \cdot Q'[t_2|t_3](\varnothing, \varnothing) \cdot Q''[t_2 \to t_3](\varnothing, \varnothing) \tag{4.11}$$

$$+ \underbrace{Q[t_2](\varnothing, \varnothing)}_{1} \cdot \underbrace{Q'[t_2|t_3](\varnothing, \{\langle 3|\})}_{0} \cdot Q''[t_2 \to t_3](\varnothing, \varnothing) \tag{4.12}$$

$$+ \underbrace{Q[t_2](\varnothing, \varnothing)}_{1} \cdot \underbrace{Q'[t_2|t_3](\varnothing, \varnothing)}_{1} \cdot \underbrace{Q''[t_2 \to t_3](\varnothing, \{\langle 3|\})}_{M_{2,3}} \tag{4.13}$$

$$= M_{2,3} \tag{4.14}$$

The full table is below, as well as in the figure 4.2. Note that while t_2 contains $|3\rangle$ and $\langle 3|$, the value $M_{3,3}$ was already eliminated. This makes sense: the node t_2 is related to t_3, which contains the row label $|2\rangle$. In the matrix, we do not have any non-zero element in row 2 that is not in the same column as $M_{3,3}$, so a permanent term including a factor of $M_{3,3}$ is not possible. The same idea would work if t_3 contained a row not in t_2.

$P[t_2] : {}_{\mathcal{R}'}\backslash^{\mathcal{C}'}$	$\varnothing$	$\langle 3\|$
$\varnothing$	0	$M_{2,3}$
$\|3\rangle$	0	0

4.3.2.3 Recursive step $t_2, t_4 \to t_1$

Reaching the root t_1, we repeat the same process. The difference is that t_1 has two children, however everything works the same. The intersections and differences, as well as the auxiliary tables, are computed individually for each child of t_1. Then the permanent table $P[t_1]$ joins them in a subset convolution. We filled this table in the figure 4.2.

Having reached the root, the permanent of the entire matrix M is found in the entry $P[t_1](\rho(t_1), \kappa(t_1))$, i.e. the table P of the root, selecting the whole submatrix in that node. This is our final result.

Chapter 5

New Algorithm

So far, we have seen Boson Sampling and the algorithms by Clifford & Clifford[11] in chapter 3. In chapter 4, we have seen how we can efficiently compute permanents of sparse matrices using tree decompositions. In this chapter, we join these and a few other ideas together, and this will allow us to quickly generate samples from a Boson Sampling experiment where the interferometer is shallow.

In section 5.1, we review what circuit depth means and what are shallow circuits. We give a construction of shallow unitaries. In section 5.2, we prove that removing rows or columns from a tree decomposition gives a valid decomposition for a submatrix. Then in section 5.3, we use these findings to give our algorithm that adapts the sampling algorithm CC-A by Clifford & Clifford with the permanent computation CP-2 by Cifuentes & Parrilo.

The idea of our algorithm is to generate a tree decomposition of the given interferometer matrix and then look at smaller parts of this decomposition to compute permanents and thus generate samples for a shallow matrix efficiently.

5.1 Shallow Circuit

The primary constraint in this work is that the interferometer being simulated is shallow. We recapitulate here what that means.

Definition 5.1.1 (reduced circuit[18])**.** For a circuit C, its reduced circuit C_{red} is obtained by first multiplying all consecutive single-mode gates into the following or preceding multi-mode gate and then multiplying together all consecutive k-mode gates that act on the same modes.

Definition 5.1.2 (depth of a circuit[18])**.** If C is a circuit, we call its depth D the maximum number of consecutive operations applied to any mode in the reduced circuit C_{red}. If C is a circuit acting on m modes, then we call C *shallow* if $D = O(\log m)$, and we call it *deep* if $D = \Omega(\log m)$.

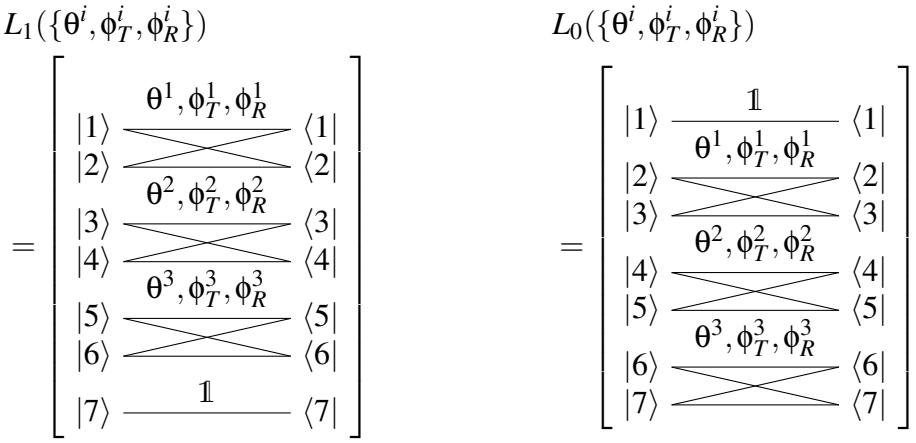


Figure 5.1: Beamsplitter layers on $m = 7$ modes

5.1.1 Beamsplitter Arrays

For us, the circuit is a unitary interferometer $\mathcal{U} \in \mathbb{U}(m)$ acting on m modes. In the original description of the problem, it was only required that $\mathcal{U}$ be a Haar random unitary; however, in this work, we will constrain $\mathcal{U}$ further to bound the depth nicely.

In section 2.5.3, we have seen the basic optical devices: the phase shifter and the beamsplitter. These two are enough to construct an arbitrary unitary transformation we like.[8, 4] Now, we define the central part of our construction. The idea is to build the interferometer up from beamsplitters, each acting on two nearest-neighbour modes. We will interleave layers of these beamsplitters acting in parallel, so that different modes may interact. Our construction was inspired by the Clements scheme[10].

Definition 5.1.3 (beamsplitter layer)**.** We define a *beamsplitter layer* $L_{0,1}(m;\{\theta^i,\phi_T^i,\phi_R^i\})$ acting on m modes, parametrised by $\theta^i,\phi_T^i,\phi_R^i$ for each constituent beamsplitter i, as

$$L_1(m;\{\theta^i,\phi_T^i,\phi_R^i\}) = B(\theta^1,\phi_T^1,\phi_R^1)\oplus\cdots\oplus B(\theta^k,\phi_T^k,\phi_R^k)\ (\oplus\mathbb{1}) \tag{5.1}$$

$$L_0(m;\{\theta^i,\phi_T^i,\phi_R^i\}) = \mathbb{1}\oplus B(\theta^1,\phi_T^1,\phi_R^1)\oplus\cdots\oplus B(\theta^k,\phi_T^k,\phi_R^k)\ (\oplus\mathbb{1}) \tag{5.2}$$

These are depicted in figure 5.1. We have two variants, which we call the *even* beamsplitter layer L_0, and *odd* layer L_1. These will allow for interleaving. The number of beamsplitters is

$$k = \begin{cases} \lfloor \frac{m}{2} - 1 \rfloor & \text{for even } m \text{ and the variant } L_0 \\ \lfloor \frac{m}{2} \rfloor & \text{otherwise.} \end{cases}$$

In (5.1) and (5.2), the $(\oplus\mathbb{1})$ is there if the number of modes requires it, so that we do not disconnect the last mode m.

Definition 5.1.4 (beamsplitter array)**.** An *alternating beamsplitter array* $\mathbf{ABA}(m,D;\{\theta^i,\phi_T^i,\phi_R^i\})$ on m modes and of depth D is a product of alternating L_0 and L_1:

$$\mathbf{ABA}(m,D;\{\theta^i,\phi_T^i,\phi_R^i\}) = \underbrace{L_{D \bmod 2}\cdots L_0\cdot L_1\cdot L_0\cdot L_1}_{D} \tag{5.3}$$

as depicted in figure 5.2. (Compare with the Clements scheme[10, 4].) This is what allows for the interleaving of nearest-neighbour interactions. We will denote the set of such possible arrays as $\mathbf{ABA}(m,D)$, i.e. without giving the angles.

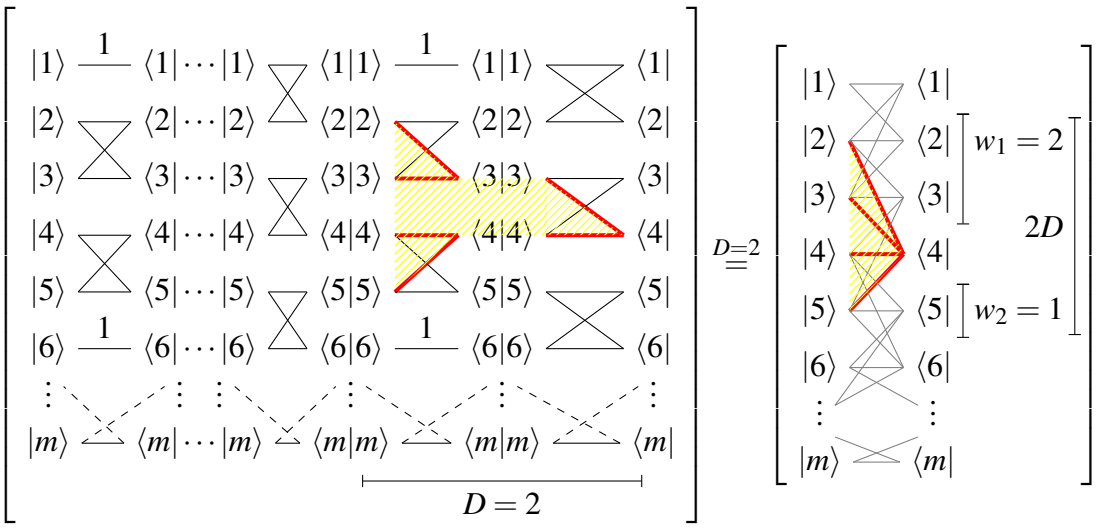


Figure 5.2: Beamsplitter array $\mathbf{ABA}(m,D;\{\theta^i,\phi_T^i,\phi_R^i\})$

Using the construction of alternating beamsplitter arrays, we can easily compose arbitrary interferometers[4] that are unitary and have not only the required depth, but also other nice properties. One of them is that $\mathcal{U} \in ABA(m,D)$ is, by construction, a banded matrix, and the depth translates directly into the bandwidths. We define what banded means below:

Definition 5.1.5 (banded matrix)**.** Let $M \in \mathbb{C}^{m\times n}$ be a matrix. We say that this is *banded*, with *bandwidths* $w_1, w_2 \in \mathbb{Z}_+$ if for all rows i and columns j, such that $i-j > w_1$ or $j-i > w_2$, the element $M_{i,j} = 0$. This creates a *band* around the main diagonal, and non-zero elements are only allowed within this band. We usually assume w_1 and w_2 are minimal. An example of a banded matrix is in figure 5.3.

Lemma 5.1.6 (bandwidth of **ABA**)**.** *If $m \in \mathbb{Z}_+$ is the number of modes, and $D \in \mathbb{Z}_+$ the chosen depth, then the matrix $\mathcal{U} \in \mathbf{ABA}(m,D)$ has bandwidth $w_1 + w_2 + 1 \leq 2D$.*

Proof. The idea is that the interleaving layers of nearest-neighbour interaction constrain how far the influence of one mode can reach; they give rise to a *causal cone*.[1] Each added layer only extends the possible reach by two new modes. This is shown for input mode (vertex) $\langle 4|$ in figure 5.2 in red.

We prove this by induction on depth D, using the graph formalism. We denote $W := w_1 + w_2 + 1$ the total width of output modes reached by an arbitrary input mode $\langle i|$.

Base case: If $D = 1$, then $\langle i|$ is connected straight across to the left to $|i\rangle$, and possibly to $|i-1\rangle$ or $|i+1\rangle$, but not both, by a beamsplitter. It may also only be connected to $|i\rangle$, if there is no beamsplitter at this position. The width is $W \leq 2 = 2D$.

[1] this is a similar concept to a *light cone* in relativity

Induction: If for $D \geq 1$ the width is $W \equiv w_1 + w_2 + 1 \leq 2D$, then we can reach $|j\rangle, \ldots, |j+W-1\rangle$ from some arbitrary $\langle i|$. The minus one is there so that we do not count past the last mode. We compose another layer $L_{0,1}$ (odd or even, as appropriate) from the left, and we denote the new widths $W' \equiv w'_1 + w'_2 + 1$.

- If $L_{0,1}$ contains a beamsplitter $\begin{bmatrix} |j-1\rangle \langle j-1| \\ |j\rangle \langle j| \end{bmatrix}$ to which $|j\rangle$ can connect, then the new width is $w'_1 = w_1 + 1$. If $L_{0,1}$ does not contain such a beamsplitter, then $w'_1 = w_1$.
- Simirarly, if $L_{0,1}$ contains a beamsplitter $\begin{bmatrix} |j+W-1\rangle \langle j+W-1| \\ |j+W\rangle \langle j+W| \end{bmatrix}$ that $|j+W-1\rangle$ connects to, then $w'_2 = w_2 + 1$. Otherwise $w'_2 = w_2$.

By adding one more layer ($D' = D + 1$), we increase the reach at most by two more output modes: $W' = w'_1 + w'_2 + 1 \leq (w_1 + 1) + (w_2 + 1) + 1 = W + 2 \leq 2D + 2 = 2D'$. Hence we have $W' \leq 2D'$.

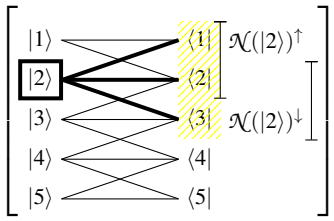
Finally, for all $D \geq 1$, we have the width $W = w_1 + w_2 + 1 \leq 2D$. Note that we have shown this in the direction from input to output modes (right-to-left in figure 5.2). The same holds in the other direction, by symmetry. ■

5.1.2 Decomposing a Banded Matrix

Now that we have a band, we use the result of Cifuentes and Parrilo: they give the optimal tree decomposition for a bipartite graph of a banded matrix in [9]. We present it in lemma 5.1.7. In figure 5.3, we give an example of a matrix with bandwidths $w_1 = w_2 = 1$, its graph representation, and its tree decomposition . We believe it is best to see the idea behind this particular decomposition visually.

$$M = \begin{pmatrix} M_{1,1} & M_{1,2} & 0 & 0 & 0 \\ M_{2,1} & M_{2,2} & M_{2,3} & 0 & 0 \\ 0 & M_{3,2} & M_{3,3} & M_{3,4} & 0 \\ 0 & 0 & M_{4,3} & M_{4,4} & M_{4,5} \\ 0 & 0 & 0 & M_{5,4} & M_{5,5} \end{pmatrix}$$

$G(M) =$



$t \in T$	$\rho(t)$	$\kappa(t)$
t_0	$\|1\rangle$	$\langle 1\|, \langle 2\|$
t_1	$\|2\rangle$	$\langle 1\|, \langle 2\|$
t_2	$\|2\rangle$	$\langle 2\|, \langle 3\|$
t_3	$\|3\rangle$	$\langle 2\|, \langle 3\|$
t_4	$\|3\rangle$	$\langle 3\|, \langle 4\|$
t_5	$\|4\rangle$	$\langle 3\|, \langle 4\|$
t_6	$\|4\rangle$	$\langle 4\|, \langle 5\|$
t_7	$\|5\rangle$	$\langle 4\|, \langle 5\|$

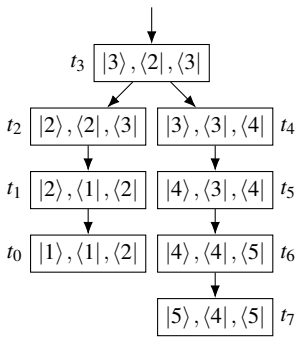


Figure 5.3: Example of tree decomposition of a banded matrix (with $w_1 = w_2 = 1$)

We provide an example in figure 5.3. We write $\mathcal{N}(|i\rangle)$ for the neighbour set of the vertex $|i\rangle$. Let $\mathcal{N}(|i\rangle)^{\uparrow}$ be the set of neighbours with the last one removed. Dually, let $\mathcal{N}(|i\rangle)^{\downarrow}$ as the set of neighbours except the first one. We depict them in the figure. The tree decomposition in lemma 5.1.7 below is generated as follows: We select $|i\rangle$ and its $\mathcal{N}(|i\rangle)^{\uparrow}$ and make this a node. Then we move the right side of the selection down by one vertex to $\mathcal{N}(|i\rangle)^{\downarrow}$, and make this another node with $|i\rangle$. Then we move down the left side of the selection to $|i+1\rangle$, which makes the right hand side $\mathcal{N}(|i+1\rangle)^{\uparrow}$. We repeat this process throughout the entire graph.

Lemma 5.1.7 (tree decomposition of a banded matrix[9])**.** *If $M \in \mathbb{C}^{m\times n}$ is a banded matrix with bandwidths w_1, w_2, and $G = G(M)$ its graph, then in general*[2] *the tree decomposition $\mathrm{T} = (T, \rho, \kappa)$ of minimum treewidth is the following: The tree is a path $T = \{t_{2(1-w_1)}, \ldots, t_{-1}, t_0, t_1, \ldots, t_{2(n-w_2)-1}\}$, where t_i is connected to t_{i+1}, and its root is chosen on one of the middle nodes. The contents of the nodes are:*

$$\rho(t_{2i-1}) = \left\{ |i+w_1\rangle \right\} \qquad \kappa(t_{2i-1}) = \left\{ \langle i|, \langle i+1|, \ldots, \langle i+w_1+w_2-1| \right\}$$
$$\rho(t_{2i}) = \left\{ |i+w_1\rangle \right\} \qquad \kappa(t_{2i}) = \left\{ \langle i+1|, \langle i+2|, \ldots, \langle i+w_1+w_2| \right\}$$

The treewidth is $\mathrm{tw}(\mathrm{T}) = w_1 + w_2$*. Note that the number of nodes* $|T| = O(n)$.

As an aside, while we usually consider banded matrices with relatively low widths w_1, w_2, we may use this decomposition also in the worst case: If the matrix has no zeros, i.e. its graph is complete, we may take the band to be the entire matrix ($w_1 = m$ and $w_2 = n$). This would allow us to use the above decomposition in this case also.

We are now ready to show a central result of this work: bounds on treewidth of a shallow interferometer, as constructed in section 5.1.1.

Theorem 5.1.8 (treewidth of **ABA**)**.** *If $\mathcal{U} \in \mathbf{ABA}(m, D)$ is a matrix of an interferometer, with $D \in \mathbb{Z}_+$ the chosen depth, and $G(\mathcal{U})$ its graph, then it has treewidth*

$$\widehat{\mathrm{tw}}(G(\mathcal{U})) \leq 2D - 1.$$

Proof. By lemma 5.1.6, we know that the bandwidths of $\mathcal{U}$ are bounded by $w_1 + w_2 + 1 \leq 2D$. In lemma 5.1.7, we have the best (in general) tree decomposition T of such a banded matrix, and its treewidth is $\mathrm{tw}(\mathrm{T}) = w_1 + w_2$. Hence, the treewidth of the graph is boundes as $\widehat{\mathrm{tw}}(G(\mathcal{U})) \leq 2D - 1$. ■

5.2 Restriction of a Tree Decomposition

In this section, we define an original[3] concept that our algorithm uses. Intuitively, a restriction of the tree decomposition means deleting the unwanted row and column vertices from all tree nodes while keeping the structure intact. We define this formally as follows, and we give an example in figure 5.4.

[2] By *in general*, we mean if we assume all allowed edges exist; of course if some of them do not, there may be an even better decomposition.

[3] as far as we know

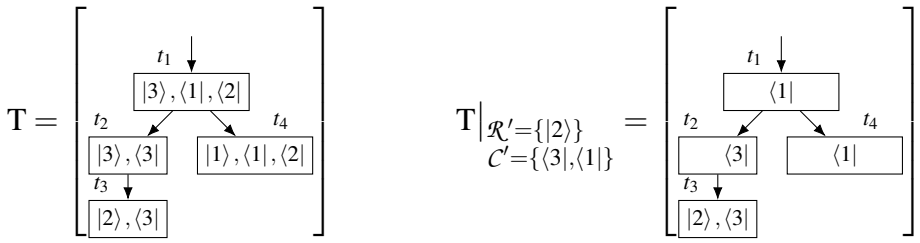


Figure 5.4: Example of a restriction of a tree decomposition

Definition 5.2.1. If $\mathrm{T} = (T, \rho, \kappa)$ is a decomposition of $G(M) = (V = (\mathcal{R}, \mathcal{C}), E)$, and $\mathcal{R}' \subseteq \mathcal{R}, \mathcal{C}' \subseteq \mathcal{C}$ are subsets of rows and columns, then the *restriction* on those subsets is $\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'} := (T, \rho', \kappa')$, with the same tree T, where ρ' and κ' are the restrictions on the ranges of ρ, κ:

$$\begin{aligned} \rho' : T &\to \mathcal{R}' & \kappa' : T &\to \mathcal{C}' \\ t &\mapsto \rho(t) \cap \mathcal{R}' & t &\mapsto \kappa(t) \cap \mathcal{C}' \quad \text{for all } t \in T \end{aligned} \tag{5.4}$$

Lemma 5.2.2. *For* $\mathrm{T} = (T, \rho, \kappa) \in \mathcal{T}(G(M))$*, the restriction* $\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'} = (T, \rho', \kappa')$ *is a tree decomposition of the submatrix* $M\big|_{\mathcal{R}',\mathcal{C}'}$*.*

Proof. We have $\mathrm{T} \in \mathcal{T}(G(M))$, and we need to check that $\mathrm{T}' \in \mathcal{T}(G(M'))$, where we denote $M' := M\big|_{\mathcal{R}',\mathcal{C}'}$. We check (T1) (we show it for ρ', but the argument works the same way for κ'):

$$\bigcup_{t \in T} \rho'(t) \overset{(5.4)}{=} \bigcup_{t \in T} \rho(t) \cap \mathcal{R}' = \left(\bigcup_{t \in T} \rho(t) \right) \cap \mathcal{R}' \overset{(\mathrm{T1})}{=} \mathcal{R} \cap \mathcal{R}' \overset{(\mathcal{R}' \subseteq \mathcal{R})}{=} \mathcal{R}' \tag{5.5}$$

and (T1) holds.

Now we check (T2). Note that the graph of a submatrix $G(M')$ is an induced subgraph of $G(M)$. This means that for all edges $(|i\rangle, \langle j|)$ in $G(M)$, if the vertices are also in $G(M')$, that is $|i\rangle \in \mathcal{R}'$ and $\langle j| \in \mathcal{C}'$, then the edge is in $G(M')$ too. By (T2) on T, for all edges $(|i\rangle, \langle j|)$, there is a node $t \in T$ such that $|i\rangle \in \rho(t)$ and $\langle j| \in \kappa(t)$. If these endpoints are in $\mathcal{R}', \mathcal{C}'$, they are in $\rho'(t), \kappa'(t)$ also, and the node t satisfies (T2) for T'. Hence (T2) holds.

Lastly, for all $|i\rangle \in \mathcal{R}'$ (resp. $\langle j| \in \mathcal{C}'$), if $\rho^{-1}(|i\rangle) := \{t \in T \mid |i\rangle \in \rho(t)\}$, the set of all nodes that contain $|i\rangle$ (resp. $\langle j|$) is a subtree of T by (T3). If the restriction includes the vertex, i.e. $|i\rangle \in \mathcal{R}'$, then by (5.4) these are exactly the nodes that contain the vertex in T'. Hence (T3) holds. ■

Lemma 5.2.3. *If we have a matrix* $M \in \mathbb{C}^{\mathcal{R} \times \mathcal{C}}$ *and the decomposition of its graph* $\mathrm{T} \in \mathcal{T}(G(M))$*, and if* $\mathcal{R}' \subseteq \mathcal{R}$ *and* $\mathcal{C}' \subseteq \mathcal{C}$ *are subsets of rows and columns of* M*, then the follwing holds:*

$$\mathrm{per}_M\left(\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'}\right) = \mathrm{per}\left(M\big|_{\mathcal{R}',\mathcal{C}'}\right) \tag{5.6}$$

where we denote $\mathrm{per}_M \mathrm{T}$ *the result of applying Algorithm CP-2 on matrix M and tree decomposition* T.

Proof. By lemma 5.2.3, the restriction $\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'}$ is a tree decomposition of $G(M\big|_{\mathcal{R}',\mathcal{C}'})$. Provided Algorithm CP-2 is correct, it will use $\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'}$ to compute the permanent of this submatrix $\mathrm{per}\, M\big|_{\mathcal{R}',\mathcal{C}'}$. ■

A restriction could yield a decomposition $\mathrm{T} = (T, \rho', \kappa')$ that has an empty node $e \in T$, i.e. $\rho'(e) = \kappa'(e) = \varnothing$. This still satisfies the axioms, but it is also compatible with permanents. We do not need a fully formal treatment. Nevertheless, the intuition is as follows: Recall from note 4.1.3 that if the decomposition is a forest, it corresponds to block-like structures in the matrix. Cutting off all connections to e (making e a disjoint tree in a forest, and likewise for every subtree to which it was connected) turns our decomposition into that same kind of forest. The permanent then becomes the product of permanents of each. From lemma 2.4.3, the permanent of no rows and no columns (so the contents of e) is 1. Hence this edge case behaves nicely.

Lemma 5.2.4 (treewidth)**.** *For a tree decomposition* T *of a graph* $G = (V = (\mathcal{R}, \mathcal{C}), E)$*, and subsets* $\mathcal{R}' \subseteq \mathcal{R}$*,* $\mathcal{C}' \subseteq \mathcal{C}$*, its restriction* $\mathrm{T}' := \mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'}$ *has treewidth*

$$\mathrm{tw}(\mathrm{T}') \leq \min\big(\mathrm{tw}(\mathrm{T}), |\mathcal{R}'| + |\mathcal{C}'| - 1\big) \tag{5.7}$$

Proof. We have $\mathrm{T} = (T, \rho, \kappa)$. If there is a node $t \in T$ such that $\mathcal{R}' \subseteq \rho(t)$ and $\mathcal{C}' \subseteq \kappa(t)$, then this must be a largest node in the restriction T′, because there can be no other vertices contained anywhere else. Hence $\mathrm{tw}(\mathrm{T}') \leq |\mathcal{R}'| + |\mathcal{C}'| - 1$, where we recall the -1 comes from the definition 4.2.1 of treewidth. A restriction can only decrease the treewidth, because it is removing vertices from nodes, so the restricted width cannot be larger than the original: $\mathrm{tw}(\mathrm{T}') \leq \mathrm{tw}(\mathrm{T})$. ■

Lemma 5.2.5 (time complexity)**.** *If we have a tree decomposition* T *of a graph* $G = (V = (\mathcal{R}, \mathcal{C}), E)$*, and subsets* $\mathcal{R}' \subseteq \mathcal{R}$*,* $\mathcal{C}' \subseteq \mathcal{C}$*, then computing the restriction* $\mathrm{T}\big|_{\mathcal{R}',\mathcal{C}'}$ *takes time* $O\big(|T| \cdot \boldsymbol{\tau}_\cap[\omega, \ell]\big)$*, where* $\omega := \mathrm{tw}(\mathrm{T})$ *and* $\ell := |\mathcal{R}'| + |\mathcal{C}'|$*, and* $\boldsymbol{\tau}_\cap$ *is the representation-dependent time complexity of set intersection as a function of set sizes.*

Proof. We have to look at every $t \in T$, which gives the factor $|T|$. For each t, we need to compute two set intersections: $\rho(t) \cap \mathcal{R}'$, and $\kappa(t) \cap \mathcal{C}'$. The sizes of nodes are bounded by treewidth: $|\rho| \leq |\rho(t)| + |\kappa(t)| \leq \omega + 1$, and likewise $|\kappa(t)| \leq \omega + 1$. The sizes of the restricting subsets are also bounded: $|\mathcal{R}|' \leq |\mathcal{R}'| + |\mathcal{C}'| = \ell$, and likewise $|\mathcal{C}'| \leq \ell$. Then we may bound the time needed for each intersection as $O(\boldsymbol{\tau}_\cap[\omega, \ell])$. ■

5.3 The Algorithm

As we hinted at the beginning of this chapter, the idea of our original proposed algorithm is to take the original matrix describing the interferometer $\mathcal{U} \in \mathbb{U}(m)$, compute its tree decomposition $\mathrm{T} = (T, \rho, \kappa)$, and then use this to sample photons, by only looking at

the relevant restrictions of T at each step. We give the pseudocode in Algorithm $\star$ (page 35).[4] Note that we are working with the original Boson Sampling from [2], where collisions of photons (i.e. two or more photons sharing an output mode) are forbidden by definition.

This is the finale where we put everything seen in previous chapters and sections together: Clifford and Clifford's algorithm (see Algorithm CC-A) tells us how to properly use the marginal probability distributions to obtain a correct sample. In that algorithm, we often need to compute permanents of ever-larger matrices. We extend the algorithm CC-A, also using the permutation trick from CC-B[5] (see sec. 3.3.1) in the following way:

We will replace the permanent calculation by algorithm CP-2 by Cifuentes and Parrilo. As seen in lemma 5.1.6, our construction of shallow interferometers gives us a banded matrix, and from lemma 5.1.7, we know how to get good tree decompositions for those. Lastly, we have lemma 5.2.3, showing that restricting a tree decomposition will give us the permanent of a submatrix, which allows us to use the easy decomposition of a banded matrix.

Algorithm $\star$: Boson Sampling using tree decomposition

Input:

- $m, n \in \mathbb{Z}_+$ such that $m > n$,
- a matrix $\mathcal{V} = \Gamma_n^{\text{in}}(\mathcal{U}) \in \mathbb{C}^{m \times n}$, which is the first n columns of $\mathcal{U} \in \mathbf{ABA}(m, D)$ the matrix of an interferometer of depth D acting on m modes

Result: a single sample $\left|\Psi_{\underline{n}'}\right\rangle$

`// Preparation`

$\mathrm{T} = (T, \rho, \kappa) \leftarrow$ decompose the matrix $\mathcal{V}$ using the method in lemma 5.1.7

$\alpha \leftarrow$ a uniformly random permutation from S_n

$\kappa(t) \leftarrow \{\alpha(c) | c \in \kappa(t)\}$ for all $t \in T$ (rename all columns labels in T using α)

`// Sampling the first photon`

$w_1(i) \leftarrow |\langle i|V|\alpha(1)\rangle|^2$ for modes $i \in [m]$

sample $\psi_1 \leftarrow i$ using an (unnormalised) pmf $w_1(i)$

`// Sampling the rest`

for *photon* $k \leftarrow 2, \ldots, n$ **do**

  for *mode* $i \in [m] \setminus \{\psi_1, \ldots, \psi_{k-1}\}$ **do**

    $\mathrm{T}' \leftarrow \mathrm{T}\big|_{\mathcal{R}'=\{\psi_1,\ldots,\psi_{k-1},i\}, C'=\alpha([k])}$ restrict the decomposition

    $w_k(i) \leftarrow \mathrm{per}_{\mathcal{V}}(\mathrm{T}')$ (see Algorithm CP-2)

  sample $\psi_k \leftarrow i$ using $w_k(i)$

$\left|\Psi_{\underline{n}'}\right\rangle \leftarrow$ sort $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$ in non-decreasing order

return $\left|\Psi_{\underline{n}'}\right\rangle$

---

[4] We call it $\star$ to signify that this is the new one.

[5] We extend CC-A and not CC-B because while we take the permutation trick from the latter, CC-B has another difference from CC-A: permanents computed by Laplace expansion, which we do not use.

5.3.1 Comments on the Steps

Here, we shortly describe the steps of Algorithm $\star$. The input is, in principle, the whole interferometer matrix $\mathcal{U} \in \mathbf{ABA}(m, D)$. However, we will only ever need to use the first n columns (the only occupied input modes), so we already specify the actual input as $\mathcal{V} = \Gamma_n^{\text{in}}(\mathcal{U})$, the matrix with only the first n columns on $\mathcal{U}$. This is a convention set by Algorithm CC-A. If we did not specify that, everything would still work, we would simply have to delete those columns in the algorithm.

Notice that we decompose this matrix (technically, its graph) first, and *then* we employ the permutation trick of Clifford & Clifford from section 3.3.1. Instead of permuting the columns of $\mathcal{V}$, we use the uniformly random permutation α to relabel the $\kappa(t)$ for each node $t \in T$. Since we mostly access the matrix via the tree decomposition, this will have the same effect as first permuting the columns of $\mathcal{V}$ (see lemma 5.3.1); except if we did that, we could no longer use the easy decomposition for banded matrices. We have to keep this in mind when computing the marginal pmf for the first photon: in CC-B, this is computed from the first column of the *permuted* matrix; for us, not having permuted columns, we use the column $\alpha(1)$.

For each subsequent photon k (with $k \in \{2, \dots, n\}$), and for each possible mode $i \in [m]$ of the new photon, we restrict the decomposition T to row labels $\mathcal{R}' = \{\psi_1, \dots, \psi_{k-1}, i\}$, i.e. all the previously sampled photon modes, and the new possibility i. The columns will be $\alpha([k]) = \{\alpha(c) | c \in [k]\}$, the first k columns, but permuted (see above).

At the end, we have the sampled photon modes $\psi_1, \dots, \psi_n$, corresponding to state $|\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$; this is just different notation. Finally, we sort them in non-decreasing order to get $\left|\Psi_{\underline{n}'}\right\rangle$ from equation (2.15), which corresponds to the the output number state $|\underline{n}'\rangle \simeq \frac{1}{\sqrt{n! \prod_{i=1}^m n_i'!}} \sum_{\sigma \in S_n} \sigma \left|\Psi_{\underline{n}'}\right\rangle$ (see equation (2.14). This is our full sample.

5.3.2 Correctness

Being an extension of CC-A, our algorithm does the sampling correctly, provided the permanents computed are the right ones. As seen in lemma 5.2.3, restricting the tree decomposition to compute permanents of submatrices is valid. We employ the permutation trick, but we only do it after we have already computed a tree decomposition. The last remaining step is to show that this does not introduce problems:

Lemma 5.3.1. *Let* $\mathrm{T} = (T, \rho, \kappa)$ *a tree decomposition of a graph G of matrix* $M \in \mathbb{C}^{\mathcal{R} \times \mathcal{C}}$*, and* $\alpha \in S_C$ *a permutation of column labels. Let* $\mathrm{T}' := (T, \rho, \kappa')$ *be a decomposition where we permute column labels:* $\kappa'(t) := \{\alpha(c) | c \in \kappa(t)\}$*, and let* M' *be a matrix, with elements* $M'_{r,c} := M_{r,\alpha(c)}$*. Then* T' *is a tree decomposition of the graph* M'*.*

Proof. Similarly to reasoning in lemma 2.4.5, the permutation $\alpha \in S_C$ is simply a relabeling of columns. In terms of the graph of matrix $G(M)$, it is a graph isomorphism that relabels column vertices and preserves edge weights. In T', it also relabels the same column vertices. Then T' must be a valid tree decomposition for $G(M')$. Intuition may be found in figure 5.5. ■

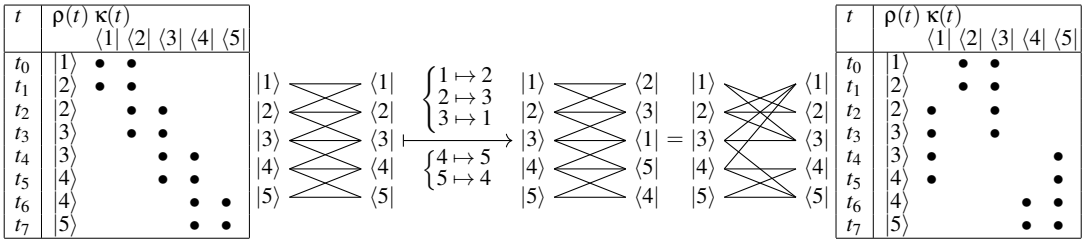


Figure 5.5: Example of a permutation acting on columns of a matrix and on its tree decomposition. On the RHS of the equality, we rearranged vertices geometrically.

5.3.3 Running Time

Lemma 5.3.2. *The time required to compute the probability for photon k with* $1 < k \leq n$ *being in mode i (see lines 8, 9 in Algorithm* $\star$*) is*

$$O\left(n + k2^{\omega'}\omega'^2\right) \tag{5.8}$$

where $\omega' \leq \min(\mathrm{tw}(\mathrm{T}), 2k-1)$ *is the treewidth of the restriction to k rows and k columns* $\mathrm{T}' := \mathrm{T}\big|_{\mathcal{R}'=\{\psi_1,\ldots,\psi_{k-1},i\},\mathcal{C}'=\alpha([k])}$.

Proof. We need to restrict the overall tree decomposition T. By lemma 5.2.5, we know that this takes time $O\big(|T| \cdot \boldsymbol{\tau}_\cap[\omega, \ell]\big)$, where $|T|$ is the number of nodes, $\omega := \mathrm{tw}(\mathrm{T})$ and $\ell := |\mathcal{R}'| + |\mathcal{C}'| = 2k$. The placeholder $\boldsymbol{\tau}_\cap$ is the time complexity of set intersection, a function of sizes of the sets being intersected. We are using a result from Cifuentes & Parrilo[9], where they assume efficient representation of sets (this could be for example a hash-map, or a bit-vector, since we know exactly what the allowed elements in our sets are). To derive time consistent with theirs, we assume the same: $\boldsymbol{\tau}_\cap = O(1)$. Our tree decomposition is constructed as in lemma 5.1.7, so the number of nodes is $|T| = \Theta(n)$. This gives the term $O(n)$ in (5.8).

Having computed the restriction, we now use Algorithm CP-2 to compute the permanent. By lemma 5.2.4, the treewidth of the restriction is $\omega' \leq \min(\mathrm{tw}(\mathrm{T}), |\mathcal{R}'| + |\mathcal{C}'| - 1) = \min(\omega, 2k-1)$. Cifuentes & Parrilo give the running time of CP-2 as $\widetilde{O}(k2^{\omega'})$ for a $k \times k$ matrix and decomposition of width ω'. Recall that $\widetilde{O}$ hides polynomial factors of ω'; however, we recover the polynomial factor from their proofs [9, Lemma 2, proof of Theorem 14], and the full running time is $O(k\omega'^2 2^{\omega'})$.

Hence, the time required to compute the value $w_k(i)$, i.e. probability of photon k being sampled in mode i is $O(n + k\omega'^2 2^{\omega'})$, where $\omega' \leq \min(\omega, 2k-1)$. ∎

Theorem 5.3.3. *The running time of Algorithm* $\star$ *for sampling n photons from an interferometer* $\mathcal{U} \in \mathbf{ABA}(m; D)$ *which is shallow, meaning* $D := c\log m = O(\log m)$ *for some chosen constant c,*[6] *is*

$$O\left(mn^3 2^{\omega'(n)}\omega'(n)^2\right) \tag{5.9}$$

[6] we introduced the constant c because the depth appears in exponents and we did not want it to disappear by just assuming $O(\log m)$

where $\omega'(n) \coloneqq O(\min(c\log m, 2n))$.

Proof. Generating the tree decomposition T (line 1) requires us to look at elements within the bands of the matrix. By lemma 5.1.6, the bandwidth of $\mathcal{U}$ is bounded as $W \coloneqq w_1 + w_2 + 1 \leq 2D$, where W is exactly the maximum number of non-zero elements in any row of the matrix. In theorem 5.1.8, we see that this is in fact related to the treewidth of our decomposition $\omega \coloneqq \mathrm{tw}(\mathrm{T}) \leq 2D - 1 = W - 1$. Our matrix has m rows (output modes), so to generate the tree decomposition, we need to look at $O(m\omega)$ elements of $\mathcal{U}$.

The first photon is sampled in time $O(m)$ because we need to compute $w_1(i)$ for $i \in [m]$, and each of these requires looking at one element of $\mathcal{U}$. We assume constant time look-up in the permutation α because this can be represented as an array $[n]^n$.

From lemma 5.3.2, we know that the time required to compute $w_k(i)$ for some $k \in \{2,\ldots,n\}$ and $i \in [m]$ is $O(n + k2^{\omega'}\omega'^2)$, where $\omega'(k) \leq \min(\omega, 2k-1)$. Sampling from $w_k(i)$ takes $O(m)$ as before. We do this for each photon and each mode, giving us

$$\sum_{k=2}^{n} m \cdot O\left(n + k2^{\omega'(k)}\omega'(k)^2\right) + O(m) = O\left(\sum_{k=2}^{n} m \cdot \left(n + k2^{\omega'(n)}\omega'(n)^2\right) + m\right) \tag{5.10}$$

$$= O\left(mn^3 2^{\omega'(n)}\omega'(n)^2 + mn^2 + mn\right) \tag{5.11}$$

The sequence summed over is increasing, so in (5.10) we give an upper bound by (pessimistically) summing over the last (largest) element. We have $D \coloneqq c\log m$ and $\omega \leq 2D - 1$, hence $\omega \leq 2c\log m - 1$, so $\omega'(n) = O(\min(c\log m, n))$. Finally, keeping only the largest growing term, the total time is

$$\begin{aligned} &O(m\omega) + O(m) + O\left(mn^3 2^{\omega'(n)}\omega'(n)^2 + mn^2 + mn\right) \\ &= O\left(m\left(n^3 2^{\omega'(n)}\omega'(n)^2 + n^2 + n + c\log m + 1\right)\right) = O\left(mn^3 2^{\omega'(n)}\omega'(n)^2\right) \end{aligned} \tag{5.12}$$

■

Theorem 5.3.4. *If we further set* $m = n^2$*, then the running time is*

$$O\left(n^{5+c}c^2\log^2 n\right). \tag{5.13}$$

Proof. Following from theorem 5.3.3, we set $m = n^2$ and simplify. We simplify first the exponent $\omega'(n) \coloneqq O(\min(c\log m, 2n)) = O(\min(2c\log n, 2n)) = O(c\log n)$. Then we simplify (5.9):

$$O\left(n^2 n^3 2^{O(c\log n)} O(c\log n)^2\right) = O\left(n^{5+c}c^2\log^2 n\right) \tag{5.14}$$

■

Chapter 6

Conclusion

We have explored ideal collision-less Boson Sampling and its classical simulation. Independently, we learned about tree decompositions and how they can be used to simplify even NP-hard problems (or, in our case of permanent computation, a #P-hard problem) by leveraging additional structure, such as sparsity.

We gave the original Algorithm $\star$ that can generate a sample from the Boson Sampling experiment for a shallow interferometer in time $O\big(n^3\, 2^{\omega'(n)}\, \omega'(n)^2\big)$, where n is the number of photons in the sample, m is the number of modes, and $\omega'(n) \coloneqq O(\min(c\log m, 2n))$. We can choose the constant c as we like: it is a parameter relating the depth of the circuit to the number of modes: $D \coloneqq O(c\log m)$. If we further set $m = n^2$, as in common, then the running time becomes $O\left(n^{5+c}\, c^2\, \log^2 n\right)$.

This running time is much lower than the general method that samples from arbitrary interferometers. For comparison, the algorithms CC-A and CC-B by Clifford & Clifford[11] take, respectively, time $O(mn\,3^n)$ and $O(n\,2^n + mn^2)$; i.e. they scale exponentially.

To obtain our algorithm, we augmented the algorithm CC-A. In fact, we based our work on an algorithm halfway between CC-A and CC-B. The latter gave us a nice trick of permuting columns to decrease complexity. However, CC-B uses Laplace expansions to compute permanents, and we needed to replace the permanent computation entirely; hence the main basis was the simpler CC-A.

We leveraged the sparsity structure of the interferometer, and we gave a construction of arbitrary unitary circuits, where we may freely choose the depth. To do this, we used our new notation for graphs of matrices which we defined as a helpful tool along the way. We proved that our construction leads to a banded matrix. Following that result, we used the Algorithm CP-2 given by Cifuentes & Parrilo[9] that can efficiently compute the permanents of sparse matrices (such as our banded interferometer matrix) using bipartite tree decompositions.

We proved new results about these decompositions, which allowed us to first decompose the whole interferometer matrix, and then use restrictions of the decomposition to compute permanents of submatrices. This result is useful because finding a tree

decomposition is, in general, difficult. However, using our method, we can instead take an easily constructed decomposition (banded matrix), and obtain from it valid decompositions of submatrices.

We have fulfilled the aims stated in section 1.1: we have understood ideal Boson Sampling and its classical simulation; tree decompositions, and the permanent algorithm based on them. We arrived at the aforementioned efficient algorithm for sampling from a shallow interferometer. For now, we are satisfied with it; however, we believe it can be made more efficient, and we give the ideas in the following section.

6.1 Future Work

6.1.1 Reuse of Partial Results

The Algorithm CP-2 uses dynamical programming to reuse previous partial results. The Algorithm CC-B does something similar: using Laplace expansion, it can compute permanents of bigger matrices from previously computed permanents of smaller submatrices.

Our algorithm is not capable of this. At each step, the decomposition that we use is different by virtue of restricting the overall decomposition of the whole interferometer matrix. For a node t of the tree, the table Q only knows about permanents of submatrices corresponding to the row and column labels contained in t; this may be reused. However, the table $P[t]$ contains permanents relating to the entire subtree T_t. When the contents of the descendant nodes change, such as when we use a restriction that allows more rows and columns, it is no longer clear if whatever we compute in $P[t]$ can be reused in subsequent steps (more photons being sampled). We believe this can be done, but we do not yet know how, so we state the following:

Conjecture 6.1.1. *It is possible to extend our Algorithm $\star$ so that the dynamic programming table $P[t]$ computed for photon k (where $k = 2, \ldots, n-1$) can be reused for the next photon $k+1$.*

6.1.2 Allowing Collisions

In this work, we assumed that no collisions of photons may happen. Such is the standard definition of Boson Sampling by [2]. However, since it was defined, researchers have generalised this problem to allow for collisions.

Our algorithm cannot handle collisions. A collision of photons means that some row is in the matrix is copied. Equivalently, a row (output) vertex in the graph of the matrix is copied, along with its edges. If that row vertex had more than one neighbour, both copies have those same neighbours, and we created a new cycle in the graph. This means the original tree decomposition is no longer valid. We either have to compute a new one that will have higher treewidth, or figure out another way to handle copying.

Conjecture 6.1.2. *It is possible to extend Algorithm $\star$ to handle collisions of photons.*

Bibliography

[1] Scott Aaronson. A linear-optical proof that the permanent is #P-hard. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 467(2136):3393–3405, 2011.

[2] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, page 333–342, New York, NY, USA, 2011. Association for Computing Machinery. (extended abstract for [3]).

[3] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9(4):143–252, 2013.

[4] B. A. Bell and I. A. Walmsley. Further compactifying linear optical unitaries. *APL Photonics*, 6(7):070804, 2021.

[5] Arjun Berera and Luigi Del Debbio. *Quantum Mechanics*. Cambridge University Press, 2021.

[6] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: Fast subset convolution. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 67–74, New York, NY, USA, 2007. Association for Computing Machinery.

[7] 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, Aug 1989.

[8] Jacques Carolan, Christopher Harrold, Chris Sparrow, Enrique Martín-López, Nicholas J. Russell, Joshua W. Silverstone, Peter J. Shadbolt, Nobuyuki Matsuda, Manabu Oguma, Mikitaka Itoh, Graham D. Marshall, Mark G. Thompson, Jonathan C. F. Matthews, Toshikazu Hashimoto, Jeremy L. O'Brien, and Anthony Laing. Universal linear optics. *Science*, 349(6249):711–716, 2015.

[9] Diego Cifuentes and Pablo A. Parrilo. An efficient tree decomposition method for permanents and mixed discriminants. *arXiv e-prints*, page arXiv:1507.03046, July 2015.

[10] William R. Clements, Peter C. Humphreys, Benjamin J. Metcalf, W. Steven Kolthammer, and Ian A. Walmsley. Optimal design for universal multiport interferometers. *Optica*, 3(12):1460–1465, Dec 2016.

[11] Peter Clifford and Raphaël Clifford. The classical complexity of boson sampling. In *SODA*, 2018.

[12] Peter Clifford and Raphaël Clifford. Faster classical boson sampling. *arXiv preprint arXiv:2005.04214*, 2020.

[13] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, 1939.

[14] Raúl García-Patrón. *Quantum information with optical continuous variables: from Bell tests to key distribution*. PhD thesis, Université libre de Bruxelles, Faculté des sciences appliquées – Informatique, Brussels, 2007.

[15] Raúl García-Patrón, Jelmer J. Renema, and Valery Shchesnovich. Simulating boson sampling in lossy architectures. *Quantum*, 3:169, August 2019.

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

[17] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

[18] Richard Jozsa. On the simulation of quantum circuits. *arXiv: Quantum Physics*, 2006.

[19] Henryk Minc. *Permanents. With a foreword by Marvin Marcus*, volume 6 of *Encycl. Math. Appl.* Cambridge University Press, Cambridge, 1978.

[20] Alexandra E. Moylett and Peter S. Turner. Quantum simulation of partially distinguishable boson sampling. *Phys. Rev. A*, 97:062329, Jun 2018.

[21] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[22] Changhun Oh, Youngrong Lim, Bill Fefferman, and Liang Jiang. Classical simulation of boson sampling based on graph structure, 2021.

[23] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[24] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[25] Gerrod Voigt. Tree decompositions, treewidth, and NP-hard problems, 2016.

[26] Herbert S. Wilf. A mechanical counting method and combinatorial applications. *Journal of Combinatorial Theory, Series A*, 4:246–258, 1968.

Appendix A

Supplementary Material

A.1 Beamsplitter

In section 2.5.3, we defined the *beamsplitter* and said that it is unitary according to our definition. Here we provide the proof. This is well known (see e.g. [7]), but we use it as a demonstration of the graph formalism and our notation 2.3.1.

Lemma A.1.1. *A beamsplitter $B(\theta,\phi_T,\phi_R)$ from definition 2.5.3 is unitary.*

Proof. We denote $B \equiv B(\theta,\phi_T,\phi_R)$, and $B^\dagger$ its adjoint.

$$B^\dagger B = \left[\begin{array}{cccc} & |i\rangle & \langle i|i\rangle & \langle i| \\ -e^{i\phi_R}\sin\theta & \diagup\diagdown & e^{-i\phi_T}\cos\theta \quad e^{i\phi_T}\cos\theta & \diagup\diagdown & -e^{-i\phi_R}\sin\theta \\ e^{-i\phi_R}\sin\theta & \diagdown\diagup & & \diagdown\diagup & e^{i\phi_R}\sin\theta \\ & |j\rangle & \langle j|j\rangle & \langle j| \end{array}\right] \tag{A.1}$$

$$= \left[\begin{array}{ccc} & |i\rangle & \langle i| \\ e^{-i\phi_R+i\phi_T}\sin\theta\cos\theta & \diagdown & e^{-i\phi_T+i\phi_R}\sin\theta\cos\theta \\ -e^{i\phi_T-i\phi_R}\sin\theta\cos\theta & & -e^{i\phi_R-i\phi_T}\sin\theta\cos\theta \\ & |j\rangle & \langle j| \end{array}\right] \tag{A.2}$$

$$= \left[\begin{array}{ccc} |i\rangle & \overset{1}{\overset{\frown}{\underset{\smile}{\quad}}} & \langle i| \\ |j\rangle & & \langle j| \end{array}\right] = \mathbb{1} \tag{A.3}$$

Likewise, $BB^\dagger = \mathbb{1}$. ∎

A.2 Greedy Algorithm

A basic tree decomposition algorithm for a graph $G = (V,E)$ that generates a decomposition works as follows:

We will generate a chain of tree nodes with no branching. Let W be a set of vertices, a selection. We initialise it with a single arbitrary vertex $v \in V$, so $W = \{v\}$. We find the set N of all neighbours of all vertices in the selection W:

$$N := \{n \in V | \exists w \in W.\ n \sim w\}$$

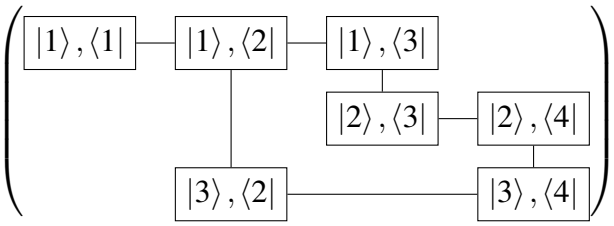
where $n \sim w$ means there is an edge between n and w. In the first step, this is just all neighbours of v. We create a node t in the tree decomposition that contains vertices $W \cup N$. Then we delete (or otherwise mark as no longer usable) vertices in W from G and set $W := N$. We repeat the process until there are no more vertices left in V.

A.3 By Decycling of the Line Graph

This sketch of an idea follows from the geometric intuition of a graph of a matrix. Let $M \in \mathbb{F}^{3\times 4}$ be some matrix

$$M := \begin{pmatrix} \bullet & \bullet & \bullet & 0 \\ 0 & 0 & \bullet & \bullet \\ 0 & \bullet & 0 & \bullet \end{pmatrix}.$$

We denote non-zero elements by $\bullet$. We represent this as a graph, and we put the edges inside the matrix. We connect the ones that share an endpoint: this is a line graph of the original matrix graph.



Now the idea is to find the cycles in this graph, and for each one, find a good place to cut it. To preserve axiom (T3), we have to find the vertex that violates (T3) and propagate it througout the uncut part of the former cycle.