Research project report

Optimization of cluster state generation for MBQC in photonics

Carried out by :

Hector Blondel
Noé Bosc-Haddad
Raphaël Fourquet
Etienne Lefranc
Emile Saint-Girons
Arman Saulière

Under the supervision of:

Zeno Toffano (CentraleSupélec, L2S)
Benoît Valiron (CentraleSupélec, LMF)
Paul Hilaire (Quandela)
Nicolas Heurtel (Quandela, CentraleSupélec)

January 19, 2025

Contents

Introduction

Context of quantum computing

In classical computing, information is processed using bits, which can be either 0 or 1. In quantum computing, information is processed using qubits, which can be in multiple states simultaneously. This allows quantum computers to perform certain calculations much faster than classical ones and this could make operations beyond the capabilities of the best supercomputers currently in use accessible.
To date, several categories of quantum objects and phenomena have been tested in order to develop the first quantum computer: trapped ions, superconductors, nuclear magnetic resonance and photons, among others. Our partner Quandela is interested in the latter particles, which are like 'grains of light'. The company intends to create quantum computers by connecting its single-photon generators to photonic circuits made with simple optical fibres.

Purpose of the research project

The calculations performed by such an optical computer would require obtaining "intricate" states, i.e. configurations involving several photons, but which cannot be described mathematically as the superposition of distinct states associated with each photon. These configurations, which take the form of an array of photons entangled in pairs, are called "cluster states". We proposed to determine an efficient process for their creation, by means of a "fusion" operation which we simulated with the Python module Perceval [3].

Structure of the report

The report is structured accordingly to the chronological order of our work. Chapter 1 shows how we got familiar with quantum computing with optical tools and contains the essential knowledge one should have to understand the rest of the project. Chapter 2 presents MBQC, fusion gates and their behavior with optical errors. Chapter 3 presents our research in creating an optimal fusion algorithm for obtaining cluster states.

Chapter 1

Basics of linear optical quantum computing

1.1 Basics of quantum computing

While called bits (or cbits) in classical computing, elementary units of information are called qubits in quantum computing.

1.1.1 Quantum states

A qubit is represented by a vector in a 2-dimensional Hilbert space. The vectors of the basis are often denoted $|0\rangle = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right)$ and $|1\rangle = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right)$. Then it could be physically implemented by any 2-dimensional quantum state e.g. the spin of an atom or an ion, the polarisation of a photon or, as described in section 1.2.1, the position of a photon. The general state of a qubit is then a superposition $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$ with $\alpha, \beta \in \mathbf{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$ A n-qubit state is a tensor product of n qubits. It is thus a vector of a 2^n-dimensional Hilbert space.

1.1.2 Quantum gates

To compute using qubits we can use quantum gates, much like classical gates they enable us to manipulate the state of one or more qubits. These gates are represented by matrices of size 2^n where n is number of Qubits the gate acts on. They have to convert any normalized state into an other normalized state and must then be unitary matrices. The basic 1 qubit gates are the following:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}, X = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}, Z = \begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$$

1.2 Linear optics

In linear optics, each qubit is represented by a photon. Applying a quantum gate involves passing one or more photons through a series of optical devices (beam plitters, mirrors, etc.).

1.2.1 Spatial mode encoding

We consider two modes (two optical fibers for example) 0 and 1. We will say that a qubit will be in the state $|0\rangle$ (resp. $|1\rangle$) if the corresponding photon is in the mode 0 (resp. 1). N qubits state will just be the tensor product of each single qubit state, showing that encoding N qubits, requires N photons and 2N modes.

Quantum gates can be implemented with linear optical tools. 50:50 beam splitters replace Hadamard gates, a mere swap of modes implements the X gate and a phase shifter on the mode 1 implements a Z gate. However, for 2-qubits gate such as the CNOT or the CZ, it is much more complicated, as it will be shown in 1.2.3.

1.2.2 Fock states

Fock states is useful when working with spatial mode encoding. Fock spaces are infinite-dimensional Hilbert spaces corresponding to a precise number of modes. However, each mode can contain any number of photons. For two modes, we will denote by $|i, j\rangle$ the state where there are i photons in the mode 0 and j photons in the mode 1. It is interesting to remark that $|0\rangle \equiv |1,0\rangle$ where the left-hand side ket denotes a quantum logical state and the right-hand side ket denotes a physical optical state. We have also $|1\rangle \equiv |0,1\rangle$. Also, while the state $|2,0\rangle$ is perfectly fine in physics, it does not make any sense in quantum computing.

We can also express each Fock state in terms of creation operators : $|i,j\rangle = \frac{(\hat{a}_0^\dagger)^i(\hat{a}_1^\dagger)^j}{\sqrt{i!}\sqrt{j!}}|0,0\rangle$. Creation operators will now be our computational basis as illustrated in 2.1.1.1.

For M modes, Fock states will also be expressed in terms of M creation operators.

1.2.3 An example : Ralph's CNOT gate

The main 2-qubit gate in quantum computing is the CNOT gate. While 1-qubit gates are deterministic in linear optics, it has been proven that 2-qubit gates cannot be deterministic when using optical tools. Thus, we can only use probabilistic gate. The simplest CNOT gate in optics is Ralph's CNOT gate (see figure 1.1). The associated unitary matrix is displayed in equation 1.1.

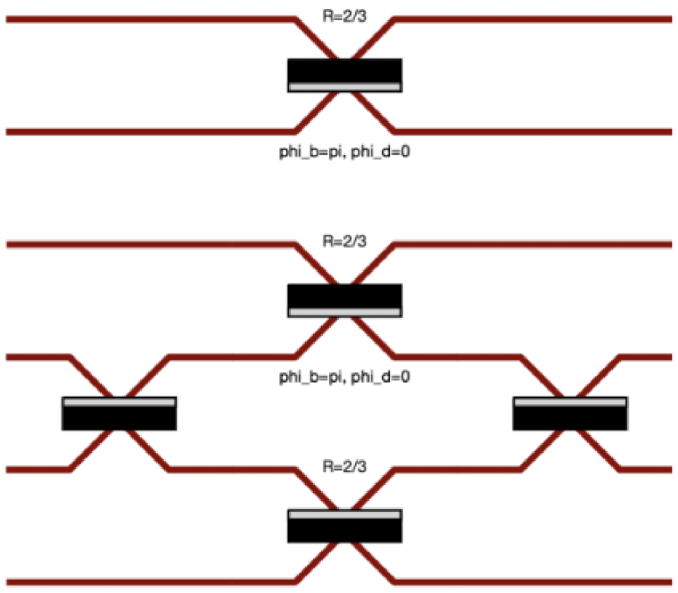


Figure 1.1: Ralph CNOT gate. The black rectangles with the grey line at the top are 50:50 beam splitters whereas the ones with the grey line at the bottom are beam splitters with 2/3 reflection coefficient and additional phase-shifters. Modes 1 and 2 contain the control qubit, modes 3 and 4 contain the target qubit, mode 0 and 5 are unoccupied ancillary modes.

$$CNOT_{Ralph} = \begin{pmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{6}i}{3} & 0 & 0 & 0 & 0 \\ -\frac{\sqrt{6}i}{3} & \frac{\sqrt{3}}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}i}{3} & -\frac{\sqrt{3}i}{3} & 0 \\ 0 & 0 & -\frac{\sqrt{3}i}{3} & \frac{\sqrt{3}}{3} & 0 & \frac{\sqrt{3}}{3} \\ 0 & 0 & -\frac{\sqrt{3}i}{3} & 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \\ 0 & 0 & 0 & \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} \end{pmatrix} \tag{1.1}$$

One can show that for any $|\psi\rangle = a|0,1,0,1,0,0\rangle + b|0,1,0,0,1,0\rangle + c|0,0,1,1,0,0\rangle + d|0,0,1,0,1,0\rangle$ such that $|a|^2+|b|^2+|c|^2+|d|^2 = 1$, then $CNOT_{Ralph}|\psi\rangle \rightarrow \frac{1}{3}(a|0,1,0,1,0,0\rangle + b|0,1,0,0,1,0\rangle + c|0,0,1,0,1,0\rangle + d|0,0,1,1,0,0\rangle) + \frac{2\sqrt{2}}{3}|\phi\rangle$ where $|\phi\rangle$ has either one photon in the ancillary modes or two photons in one mode. This shows that the success rate of the Ralph's CNOT is $\frac{1}{9}$.

1.3 An application : Quantum Teleportation

1.3.1 Presentation of the process

The aim of quantum teleportation is to take as input a qubit in an unknown state $|\psi\rangle$ and to transfer its state to another qubit located elsewhere. Effectively transporting the qubit since the qubit in input and the one in output have the exact same state.

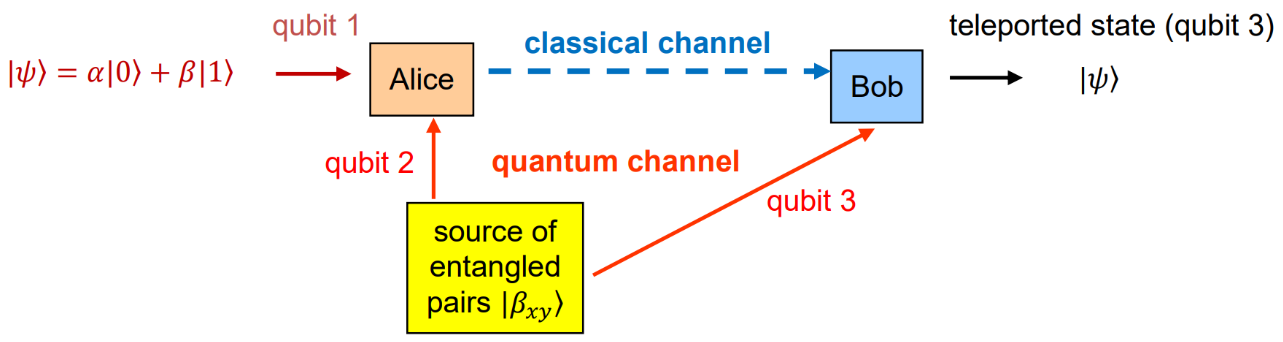

qubit 1
$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
Alice
classical channel
Bob
teleported state (qubit 3)
$|\psi\rangle$
qubit 2
quantum channel
qubit 3
source of entangled pairs $|\beta_{xy}\rangle$

Figure 1.2: Experimental setup to teleport a particle

As we see on the figure 1.2 we will need both a classical channel and a quantum channel as well as a source of entangled qubits in the Bell state $|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Which we can produce using the circuit presented in fig. 1.3.

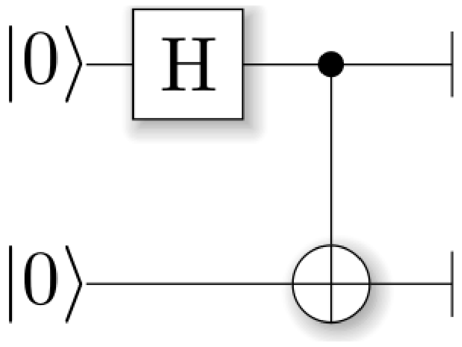


Figure 1.3: Quantum circuit to create the bell pair $|\beta_{00}\rangle$

We then complete the circuit with another CNOT followed by an Hadamard gate, giving us the circuit described by fig. 1.4.

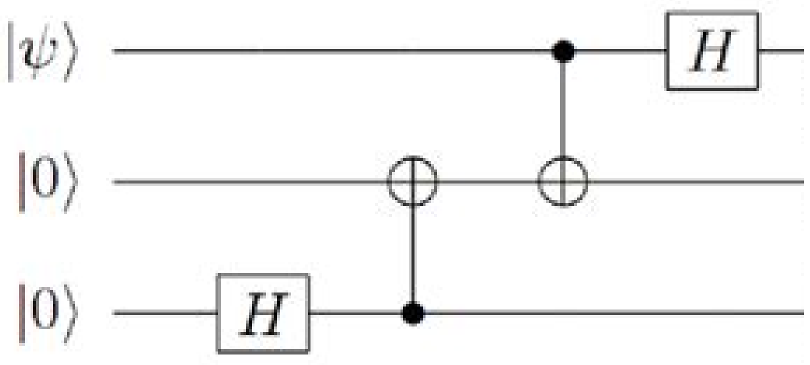


Figure 1.4: Quantum circuit to enable quantum teleportation

At this stage we have the following state for our three entangled qubits:

$$\frac{1}{2}\left[\alpha\left(|000\rangle + |011\rangle + |100\rangle + |111\rangle\right) + \beta\left(|110\rangle + |101\rangle - |010\rangle - |001\rangle\right)\right] \tag{1.2}$$

We then measure the first two qubit and send the third one to Bob through a quantum channel and apply the following gates depending on the results of the measurements.

Qubit1	Qubit2	Qubit3	Gates to apply
0	0	$\alpha\|0\rangle + \beta\|1\rangle$	I
0	1	$\alpha\|1\rangle + \beta\|0\rangle$	X
1	0	$\alpha\|0\rangle - \beta\|1\rangle$	Z
1	1	$\alpha\|1\rangle - \beta\|0\rangle$	XZ

Table 1.1: Measured value on qubit 1 and 2 and resulting gates to apply on qubit 3 to obtain $|\psi\rangle$

We have therefore the following circuit for the teleportation operation in it's entirety.

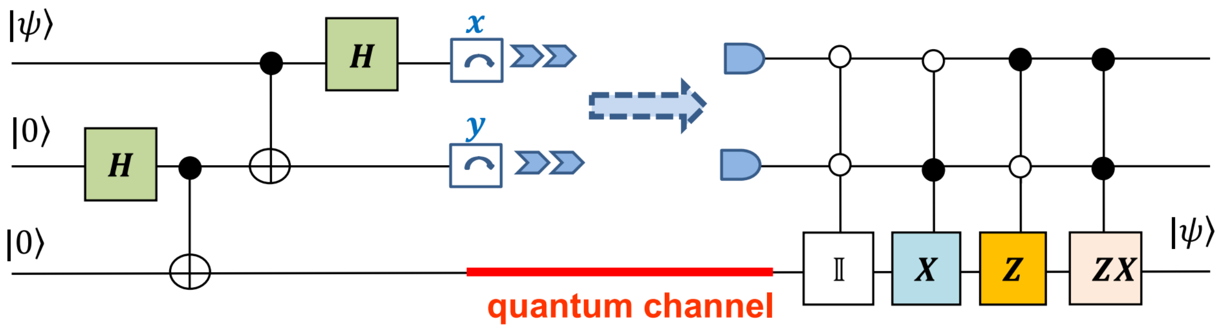


Figure 1.5: Quantum circuit to enable quantum teleportation

We can also simplify the circuit of Bob by removing Toffoli gates and having an Control X controlled by y and a Control Z controlled by X.

1.3.2 Quantum teleportation with linear optics tools

We can easily use linear optic tools to enable quantum teleportation, for it to work we can just reproduce the quantum circuits from 1.5 using the linear optics gates from 1.2 giving us the following circuits for Alice and for Bob:

We can realize that using photons is in this case quite pratical for realizing the quantum channel needed

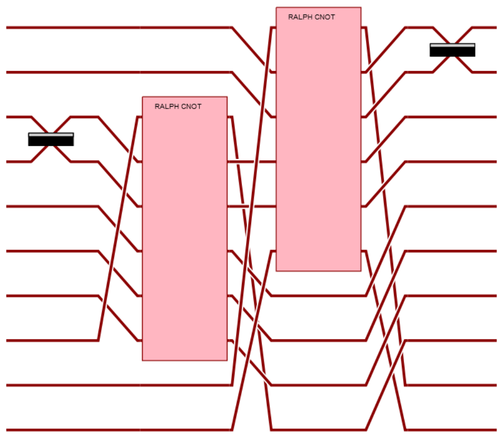


Figure 1.6: Alice linear optics circuit for teleportation

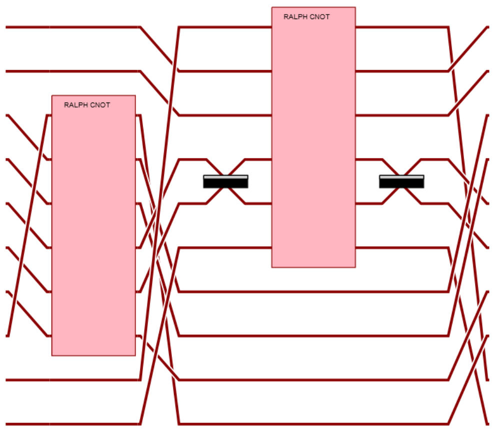


Figure 1.7: Bob linear optics circuit for teleportation

for the quantum teleportation. Indeed to transport a qubit encoded on a photon you can simply use a fiber optics. However the rest of the circuit present us with the redundant problem that two qbits quantum gates are probabilistics on photonic qubits, giving the circuit a relatively low success chance. Which we can calculate knowing the probability of success of the CNOT gate (since the C2 gate has the same success rate). The probability is therefore:

$$\text{Success\_rate} = (\text{CNOT\_success\_rate})^3 * (\text{CZ\_success\_rate}) = Cnot\_success\_rate)^4 = \frac{1}{6561} \tag{1.3}$$

Chapter 2

Towards Measurement based quantum computing

We have witnessed in section 1.3.2 how the success rate of an algorithm sinks with some naive linear optical components such as 2-qubits gates, which work rarely.

Thus, we are turning ourselves to a new way of performing quantum computing algorithms which is called MBQC (Measurement Based Quantum Computing). The main idea is to work with graphs: our algorithms will no longer be an input state and a series of gate we apply to our state in a specific order but an input cluster state (which will have qubits as vertices and entanglement as edges) and a series of measurement that will be performed in a specific order on each qubit.

In this chapter, we will focus on errors commited by single photon sourcing as well as presenting MBQC and see how we can perform it with linear optics.

2.1 Sources of error in linear optics

The three main errors one has to deal with when doing linear optics are indistinguishability, purity and brightness which are related to errors made by photon sources. Section 2.1.1 presents indistinguishability while section 2.1.2 presents purity. They will be useful in section 2.3.3.

2.1.1 Hong-Ou-Mandel effect

2.1.1.1 Indistinguishable case

The Hong-Ou-Mandel effect is when 2 indistinguishable photons pass through a beam splitter, they come out on the same side.
This is an interference experiment on a separating plate. When two photons go simultaneously on the two faces of the beam splitter, they leave it together on the same side, while four possibilities exist in principle. This phenomenon called coalescence is deeply linked to the quantum nature of photons which are bosons.

The proof of this result is using quantum physics tools. A beam splitter is in quantum computing a Hadamard-gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$ acting on creation operators. Let $\hat{a}_1^\dagger$ and $\hat{a}_2^\dagger$ be the creation operators in modes 1 and 2. We have thus that $H\hat{a}_1^\dagger = \frac{1}{\sqrt{2}}(\hat{a}_1^\dagger + \hat{a}_2^\dagger)$ and $H\hat{a}_2^\dagger = \frac{1}{\sqrt{2}}(\hat{a}_1^\dagger - \hat{a}_2^\dagger)$. The state of two photons where each one goes on one face of the beam splitter is: $|1,1\rangle = \hat{a}_1^\dagger\hat{a}_2^\dagger |0,0\rangle$. After the gate the state of the two photons is:

$$\begin{aligned}(H\hat{a}_1^\dagger)(H\hat{a}_2^\dagger)|0,0\rangle &= \frac{1}{\sqrt{2}}(\hat{a}_1^\dagger + \hat{a}_2^\dagger)\frac{1}{\sqrt{2}}(\hat{a}_1^\dagger - \hat{a}_2^\dagger)|0,0\rangle \\ &= \frac{1}{2}((\hat{a}_1^\dagger)^2 - (\hat{a}_2^\dagger)^2)|0,0\rangle \quad \text{because } [\hat{a}_1^\dagger, \hat{a}_2^\dagger] = \hat{0} \\ &= \frac{1}{\sqrt{2}}(|2,0\rangle - |0,2\rangle) \quad \text{because } \hat{a}^\dagger|1\rangle = \sqrt{2}|2\rangle\end{aligned} \tag{2.1}$$

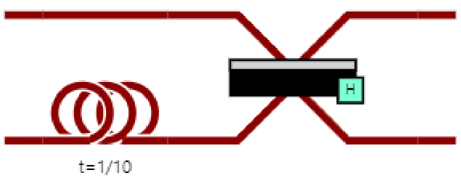


Figure 2.1: Hom experiment with time delay $\frac{1}{10}$ of the period of photons

This is indeed the result we announced, the two photons can only exit the beam splitter on the same side.

2.1.1.2 Distinguishable case

Now, we will show how this result is no longer true when we deal with partial distinguishable photons. The partial distinguishability between two photons is well described in [4]. It is defined as a function of the overlap S between the wavepackets of the two photons. One can write the wavepacket of a photon as so: $|P\rangle = \int dtK(t)e^{-i\omega(t-t_0)}\,|t\rangle$ where K is an enveloppe function, ω is the frequency of the wave and t_0 is a central time. The overlap between the wavepackets of two photons 1 and 2 is thus $S = \langle P_1|P_2\rangle = \int dtK_1(t)K_2(t)e^{i(\omega_1(t-t_{1,0})-\omega_2(t-t_{2,0}))}$.

A nice way to create partially distinguishable photons is to take a source capable of producing two indistinguishable photons and operating a time delay on one of the photons. In the case of the HOM experiment, it corresponds to the circuit 2.1. The Δt-time delay operator can be expressed as $\hat{T}_{\Delta t}\,|t\rangle = |t+\Delta t\rangle$. The two wavepackets are the same before the time delay $|P_1\rangle = |P_2\rangle = \int dtK(t)e^{-i\omega(t-t_0)}\,|t\rangle$. After the time delay, the wavepacket of the second photon become: $|P_2\rangle = \int dtK(t)e^{-i\omega(t-t_0)}\,|t+\Delta t\rangle$. The value of the overlap is then

$$\begin{aligned} S = \langle P_1|P_2\rangle &= \int_{\mathbb{R}^2} dt_1 dt_2 K(t_1)K(t_2)e^{i(\omega(t_1-t_0)-\omega(t_2-t_0))}\,\langle t_1|t_2+\Delta t\rangle \\ &= \int_{\mathbb{R}} dt_2 K(t_2+\Delta t)K(t_2)e^{i(\omega(t_2+\Delta t-t_0)-\omega(t_2-t_0))} \\ &= e^{i\omega\Delta t}\int_{\mathbb{R}} dtK(t+\Delta t)K(t) \end{aligned}$$

We can then define the indistinguishability between two photons as the ratio $I(\Delta t)$ expressed in equation 2.2.

$$I(\Delta t) = \frac{\int_{\mathbb{R}} dtK(t+\Delta t)K(t)}{||K||_2^2} \tag{2.2}$$

Assuming that K is positive, I is between 0 and 1. It is direct from 2.2 that $I(0) = 1$ which is the case where the two photons are indistinguishable. If Δt is bigger than the coherence time of the wavepacket, which mathematically means that $\Delta t > \lambda(Supp(K))$ (where λ denotes the Lebesgue measure), then $I(\Delta t) = 0$. If $Supp(K) = +\infty$, we just have $I(+\infty) = 0$.

Implementing distinguishability on Perceval is quite easy with processors. Imagining that the two photons are Gaussian-shaped with frequency $\omega = 2\pi$, standard deviation $\Delta\omega = 2\pi$ and central time are $t_0 = 0$. We can derivate $S = e^{i\Delta t 2\pi}e^{-(\Delta t)^2\pi^2}$ (derivation is done in [4]). Thus $I(\Delta t) = e^{-(\Delta t)^2\pi^2}$. We can then retrieve the result of the HOM visibility experiment done in [4]. We plot the results on figure 2.2.

$\Delta t = 0$ corresponds to the indistinguishable case and the output state is $\frac{1}{\sqrt{2}}(|2,0\rangle - |0,2\rangle)$. $\Delta t \to \infty$ corresponds to the totally distinguishable case and the output is $\frac{1}{2}(|2,0\rangle - |0,2\rangle) + \frac{1}{\sqrt{2}}\,|1,1\rangle$.

In practice, we quantify the distinguishability of wavepackets by doing a HOM experiment.

2.1.2 Hanbury-Brown-Twiss effect

The HBT experiment is useful to quantify the purity of a source. A source of photons is called pure if it produces less than one photon when asked to produce one photon. The HBT experiment consists in a photon stream arriving on one face of a 50:50 beam splitter and two detectors behind each face of the beam splitter. It is illustrated on figure 2.3.

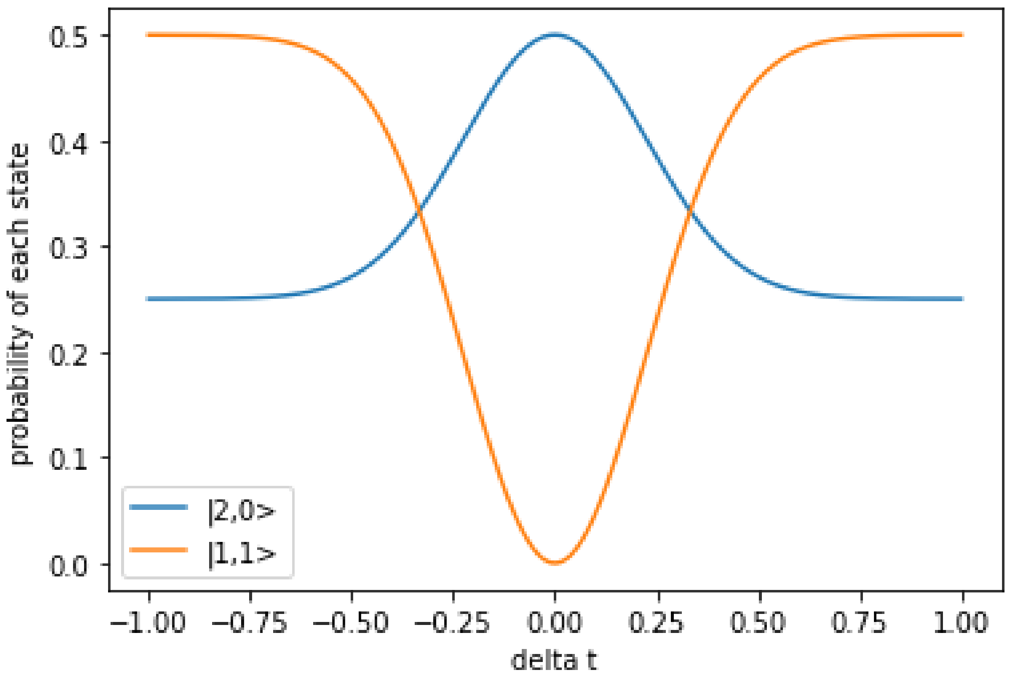


Figure 2.2: HOM visibility variations with time delay

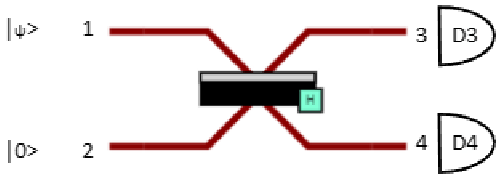


Figure 2.3: HBT experiment with photons

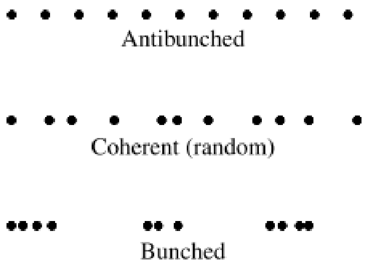


Figure 2.4: Illustration of different states of light

2.1.2.1 Second-order correlation function $g^{(2)}(0)$

We will not go through the HBT effect in classical optics. It is fully described in [1]. In quantum optics, we recall the expression of the second-order correlation function for two detectors in equation 2.3. We call modes 1 and 2 the two input modes of the beam splitter and 3 and 4 its two output modes.

$$g^{(2)}(0) = \frac{\langle \hat{n}_3 \hat{n}_4 \rangle_{|\psi,0\rangle_{1,2}}}{\langle \hat{n}_3 \rangle_{|\psi,0\rangle_{1,2}} \langle \hat{n}_4 \rangle_{|\psi,0\rangle_{1,2}}} \tag{2.3}$$

where $\hat{n}_3$ and $\hat{n}_4$ are the number of photons counted on each detectors and $|\psi\rangle_1$ is the state of the photon stream arriving on input 1. In classical optics, $g^{(2)}(0) \geq 1$ which no longer stands when dealing with single photon sources, showing quantum properties of light. We can distinguish three different sources of light:

- $g^{(2)}(0) > 1$ which means that the light is bunched: photons try to group in wavepackets just like in classical optics. We called it bunched light. It is likely that if I detect one photon with the first detector, I will detect one at the same time with the other detector.
- $g^{(2)}(0) = 1$, which caracterizes some randomness of the source, one photon will behave independently of the others. We call it coherent light. Detecting one photon with one detector will have no incidence on detecting one with the other. Hence, $\langle \hat{n}_3 \hat{n}_4 \rangle = \langle \hat{n}_3 \rangle \langle \hat{n}_4 \rangle$
- $g^{(2)}(0) < 1$ which shows that one photon will try as much possible to not group with others. If I detect one photon with the first detector, it is not likely that I will detect one at the same time with the other detector.

It is illustrated on figure 2.4 taken from [1]. It can be useful to derive from equation 2.3 an expression of $g^{(2)}(0)$ in terms of photon numbers in modes 1 and 2 only.

$$\begin{aligned}
\langle \hat{n}_3 \hat{n}_4 \rangle_{|\psi,0\rangle_{1,2}} &= \langle \hat{a}_3^\dagger \hat{a}_4^\dagger \hat{a}_3 \hat{a}_4 \rangle_{|\psi,0\rangle_{1,2}} \quad \text{since} \;\; \hat{n} = \hat{a}^\dagger \hat{a} \;\; \text{and} \;\; [\hat{a}_4^\dagger, \hat{a}_3] = \hat{0} \\
&= \frac{1}{4} \langle \psi, 0| (\hat{a}_1^\dagger + \hat{a}_2^\dagger)(\hat{a}_1^\dagger - \hat{a}_2^\dagger)(\hat{a}_1 + \hat{a}_2)(\hat{a}_1 - \hat{a}_2) |\psi, 0\rangle \quad \text{as explained in 2.1.1.1} \\
&= \frac{1}{4} \langle \psi| \hat{a}_1^\dagger \hat{a}_1^\dagger \hat{a}_1 \hat{a}_1 |\psi\rangle \quad \text{because} \;\; \hat{a}|0\rangle = \langle 0| \hat{a}^\dagger = 0 \\
&= \frac{1}{4} (\langle \psi| \hat{n}_1^2 |\psi\rangle - \langle \psi| \hat{n}_1 |\psi\rangle) \quad \text{because} \;\; \hat{a}^\dagger \hat{a} = \hat{a}\hat{a}^\dagger - \hat{\mathbb{1}} \\
\langle \hat{n}_3 \rangle_{|\psi,0\rangle_{1,2}} = \langle \hat{n}_4 \rangle_{|\psi,0\rangle_{1,2}} &= \frac{1}{2} \langle \psi| \hat{n}_1 |\psi\rangle \quad \text{with the same calculations}
\end{aligned}$$

We obtain a new expression of $g^{(2)}(0)$ in equation 2.4.

$$g^{(2)}(0) = \frac{\langle \hat{n}_1^2 \rangle_\psi - \langle \hat{n}_1 \rangle_\psi}{\langle \hat{n}_1 \rangle_\psi^2} = \frac{\langle \psi| \hat{n}_1^2 |\psi\rangle - \langle \psi| \hat{n}_1 |\psi\rangle}{(\langle \psi| \hat{n}_1 |\psi\rangle)^2} \tag{2.4}$$

One may define for a single photon source its **purity** $p \in [0, 1]$ as a decreasing function of $g^{(2)}(0)$. We keep the property that the source is pure (i.e. $p = 1$ and $g^{(2)}(0) = 0$) if and only if a photon does not come with another. You should be able to have a single photon when asking.

2.1.2.2 Examples

For example, if the photon stream is in a coherent state, which corresponds to

$$|\psi\rangle = |\alpha\rangle = e^{-\frac{|\alpha|^2}{2}} \sum_{n_1=0}^{\infty} \frac{\alpha^{n_1}}{n_1!} |n_1\rangle$$

Then, since $|\alpha\rangle$ is following a Poisson distribution of parameter $|\alpha|^2$,

$$g^{(2)}(0) = \frac{\mathbb{V}\mathrm{ar}(\mathcal{P}(|\alpha|^2)) + \mathbb{E}(\mathcal{P}(|\alpha|^2))^2 - \mathbb{E}(\mathcal{P}(|\alpha|^2))}{\mathbb{E}(\mathcal{P}(|\alpha|^2))^2} = \frac{|\alpha|^2 + |\alpha|^4 - |\alpha|^2}{|\alpha|^4} = 1$$

which justifies the name of "coherent" state.

At the contrary without any surprise, a photon stream of one photon, $|\psi\rangle = |1\rangle$ is in some sense the most "quantum" stream possible and $g^{(2)}(0) = \frac{1-1}{1} = 0$. More generally, if the photon stream is in any Fock state $|n\rangle$, we have $g^{(2)}(0) = 1 - \frac{1}{n}$, showing that the more photons you have, the greater $g^{(2)}(0)$ will be and you will get back to classical optics description.

2.2 Measurement-based quantum computation

Measurement-based quantum computing (or one way quantum computing) is a way to perform quantum computations without using gates and circuits. To do so, given a problem, you prepare your quantum system in a certain cluster state (which are described in the next section). Then, you perform a series of measurement in a certain order. These measurements will destroy some qubits while the state of the others will be modified because of entanglement. The final state will be the solution of the problem.

Cluster states

A cluster state is a graph state where each vertex is a qubit and an edge between two qubits means that they are entangled. More precisely, each qubit is in the state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and each entanglement is done by applying a CZ gate. So, given a graph state $G = (V = \{1, ..., N\}, E = \{(i,j) \in V^2 \mid i < j \text{ and there is an edge between i and j}\})$, the cluster state associated with G is given by $|G\rangle = \sum_{(i,j)\in E} CZ_{i,j} |+...+\rangle_N$.

2.3 Implementing fusion gates

MBQC relies on graph states and measurement and since measurement is quite easy to perform with spatial mode encoding, the key to perform MBQC is building cluster states which is not easy.

2.3.1 Building linear cluster states

Building linear cluster states can be done in principle with probability 1 without using fusion gates but using a quantum emitter capable of emitting entangled photons. Linear cluster states will appear to be useful when in chapter 3 we will prove that you can build any cluster state from a set of linear cluster states.

A linear cluster states of length k is a set of k qubits, each prepared in state $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. We then apply a CZ gate between each pair of qubits that are neighbors. We thus have the following state: $|\psi\rangle_k = CZ_{0,1}CZ_{1,2}...CZ_{k-1,k} |+ + ...+\rangle_k$. We can easily show that $|\psi\rangle_2 = |00\rangle + |01\rangle + |10\rangle - |11\rangle$.

In Fock states representation, we have the following: $|\psi\rangle_3 = |1,0,1,0,1,0\rangle + |1,0,1,0,0,1\rangle + |1,0,0,1,1,0\rangle - |1,0,0,1,0,1\rangle + |0,1,1,0,1,0\rangle + |0,1,1,0,0,1\rangle - |0,1,0,1,1,0\rangle + |0,1,0,1,0,1\rangle$.

To build such states with linear optics, rather than using a CZ gate such as the Knill CZ with probability success $\frac{2}{27}$ (see [5]), we will use a simple circuit which builds a superposition of a huge number of states and then we will use postprocessing (filtering) to obtain our desired state. To obtain a linear cluster state of k qubits, we will start by preparing each qubit in state $|+\rangle$ from input state $|1,0\rangle$. This is achieved by putting a beam splitter in each pair of modes. Then for each pair of neighboring qubits $(i, i+1)$, we will exchange modes 1 of the two qubits then we will place a beam splitter on the second qubit $i+1$. We obtain the circuit displayed with Perceval on figure 2.5 for $k = 3$. The output state will be then: $\frac{1}{2}|\psi\rangle_k + \frac{\sqrt{3}}{2}|\phi\rangle$ (which explicit form has been computed with Perceval and displayed in the appendices 27). The explicit form of $|\phi\rangle$

is not relevant. All we need to know is that $|\phi\rangle$ is a superposition of multiple classical states where for each of them, there exists one pair of mode $(2i, 2i+1)$ such that there are two photons in two modes affected to a single qubit, which makes no mathematical sense. We thus postprocess the output state on the property that there should be one and only one photon as a qubit for each pair of modes $(2i, 2i+1)$. We then obtain $|\psi\rangle_k$.

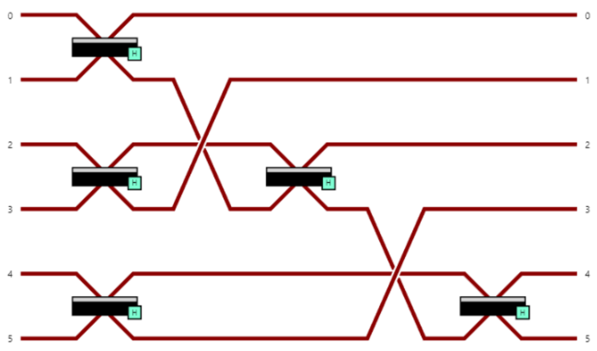


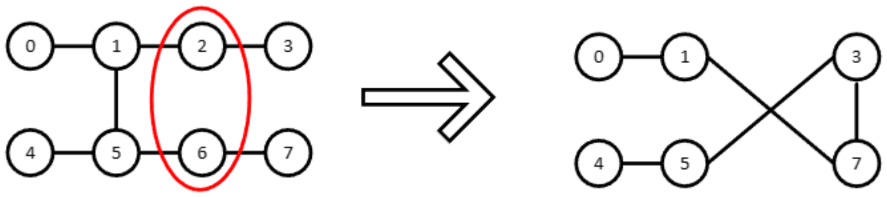
Figure 2.5: Circuit for linear cluster state of length 3

Note by P. Hilaire: *That is for example the way we did it three years ago in the lab (sort of). One of the issues of doing that is that the "non relevant" $|\phi\rangle$ becomes relevant if you are not able to post-select immediately after the gates (that you do other gates after)... In that case, we can't really concatenate them and using for other fusion gates. Fortunately, we can use deterministic generation with a quantum emitter that can be concatenated!*

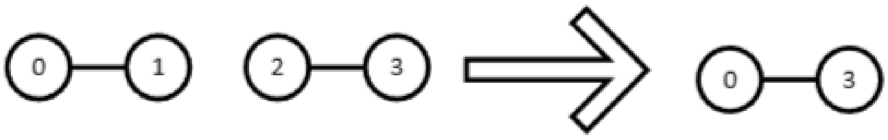
2.3.2 Fusion gates

Now that we have our linear cluster states of any length, we want to perform fusions between two cluster states to obtain more complex cluster states. This will be done with fusion gates.

We will tackle the type-II gate which is well described in [2]. A type-II gate takes a cluster state of N qubits, performs a fusion between two qubits i and j and returns a cluster state of N-2 qubits. The two qubits i and j are thus destroyed by measurement. For any neighbor v_i of i (v_i and i are entangled) and for any neighbor v_j of j, if v_i and v_j are neighbors then they will no longer be neighbors and if they were not they will become neighbors. i and j will be discarded. We give an example on figure 2.6a.



(a) Fusion between qubits 2 and 6



(b) Fusion between two linear cluster states

In linear optics those gates are done by putting a beam splitter on the modes of the first qubit, permuting modes 1 of both qubit and putting a beam splitter on the modes on each qubit. The circuit is diplayed with Perceval on figure 2.7. We measure the four modes of the two qubits (which will destroy these two qubits) and if there are not exactly one qubit in modes 0 and 1 and one qubit in modes 2 and 3, we say that the gate has failed, otherwise the gate has worked successfully. This case happens 50% of the time, we say that the success rate of the gate is $\frac{1}{2}$.

We shall check now that this gate works accordingly to the definition. We want our gate to fusion 2 cluster states of length 2 as one cluster state still of length 2. This is illustrated on figure 2.6b. And this gate should work 50% of the time. However, after computing this fusion on Perceval, we observe that the gate fails 50% of

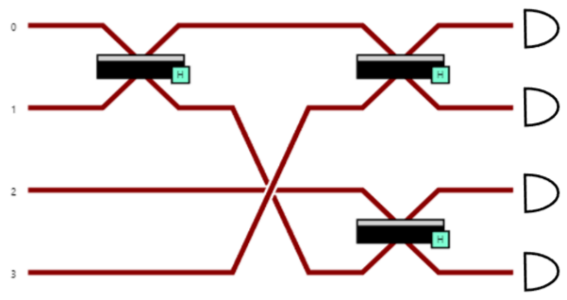


Figure 2.7: Type-II fusion gate

the time but only 25% of the time it returns our desired state. The rest of the time, it returns a state slightly different which is $(Z_0)(CZ)\left|++\right\rangle = \frac{1}{\sqrt(2)}(\left|00\right\rangle + \left|01\right\rangle - \left|10\right\rangle + \left|11\right\rangle$. However, it is not that big a deal, we can correct it without too much difficulty so we will not take it into account in the future. This is summed up in the following:

input	failure	$CZ\left\|++\right\rangle$	$Z_0(CZ)\left\|++\right\rangle$
$CZ\left\|++\right\rangle \otimes CZ\left\|++\right\rangle$	1/2	1/4	1/4

Moreover, there is a way to increase the success rate of fusion gates. This can be done by adding four ancillary photons and design a more complex circuit. This new gates are called boosted fusion gate and have a success rate of $\frac{3}{4}$. They are described as well in [2] but their study is much more complicated than the regular ones. Their circuit is given on figure 2.8. The two qubits on which we apply the fusion are on modes 0 to 3. The four ancillary photons are on modes 4 to 7.

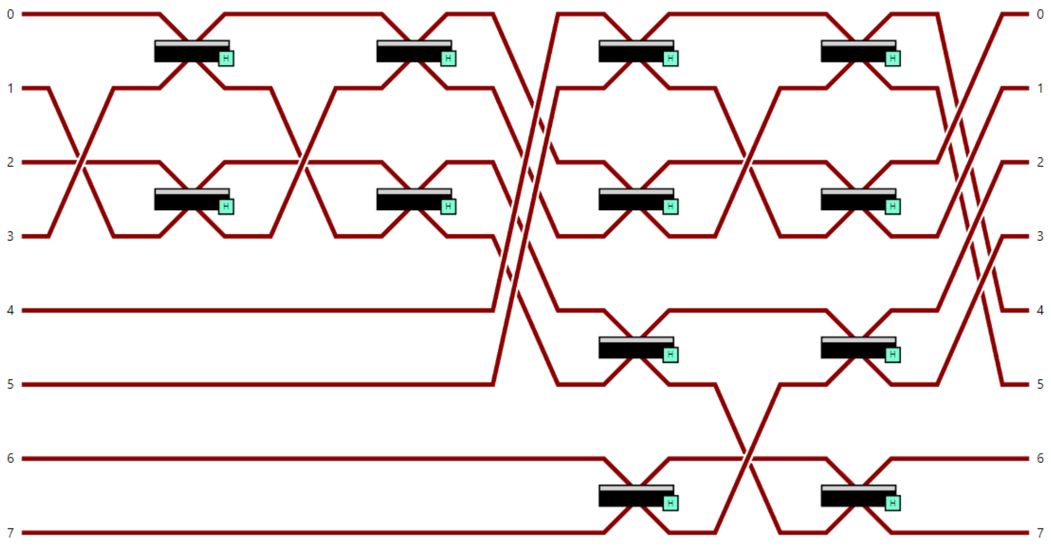
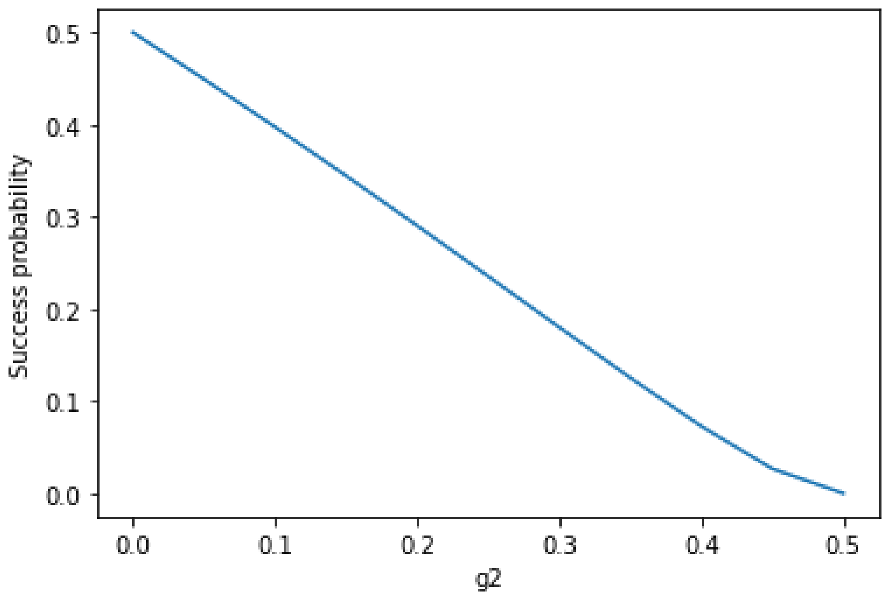


Figure 2.8: Boosted fusion gate

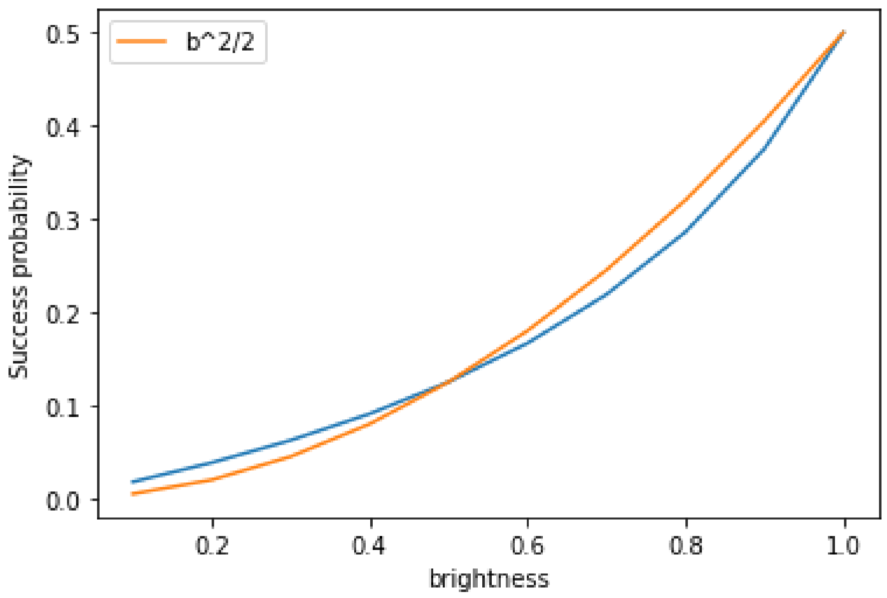
2.3.3 Fusion gates behavior with errors

We are now interested on how purity and brightness affect the success rate of our fusion gates. We mentioned purity in section 2.1.2.1. Purity characterizes how well a source can give me one photon or less when asked for one. However, we will use $g^{(2)}(0)$ (decreasing function of purity) as Perceval handles it. On the other hand, brightness characterizes how well a source can give me one photon or more when asked for one.

We can simulate on Perceval how the success rate will be modified. The way we did it was a bit basic (but provided nonetheless quite fair results) since Perceval prefers to handle classical states as inputs when computing probabilities. We take an arbitrary input classical state $\left|\phi_{in}\right\rangle$ (e.g. $\left|\phi_{in}\right\rangle = \left|1,0,1,0,1,0,1,0\right\rangle$) and compute its output after the fusion gate (only the unitary part, without performing any measurement) $\left|\psi_{out}\right\rangle$ which we can decompose as $|\psi_{out}\rangle = \sqrt{p(g^{(2)}, b)}\,|\psi_{math}\rangle + \sqrt{1 - p(g^{(2)}, b)}\,|\psi_{nonmath}\rangle$ where $|\psi_{math}\rangle$ is a superposition of states where there is at most one photon on each mode and $|\psi_{nonmath}\rangle$ contains more than two photons on at least one mode. $p(g^{(2)}, b)$ is the success rate of the fusion gate ($g^{(2)} \equiv g^{(2)}(0)$ and b corresponds to brightness). It is straightforward from 2.3.2 that $p(g^{(2)} = 0, b = 1) = \frac{1}{2}$. The computation of $p(g^{(2)}, b)$ can be done very easily on Perceval, leading to the plots on 2.9. We retrieve common results: $p(g^{(2)}, b = 1) \approx k_1.g^{(2)}(0) + k_2$ (see 2.9a) and $p(g^{(2)} = 0, b) \approx \frac{b^2}{2}$ (see 2.9b).



(a) Success rate with $g^{(2)}(0)$ (brightness=1)



(b) Success rate (blue curve) with brightness ($g^{(2)}(0) = 0$)

Figure 2.9: Success rate variations of fusion gates with $g^{(2)}(0)$ and brightness

Chapter 3

In search of the optimal fusion program for a given graph

The goal set by Quandela consisted in creating any graph from either linear qubits chains (of any length) or qubits stars (with any number of branches), whose cost is assumed to be null. These basic structures are then merged using the fusion gate above-mentioned, which can be summarised that way:

- two vertices (qubits) can be merged only if they are not neighbours and they have no common neighbour;
- merging vertices a and b leads to the deletion of a and b. The vertices of a are connected, after merging, to all the neighbours of b to which they were not connected before merging, and are disconnected from those to which they were connected (see Fig. 2.6a and 2.6b).

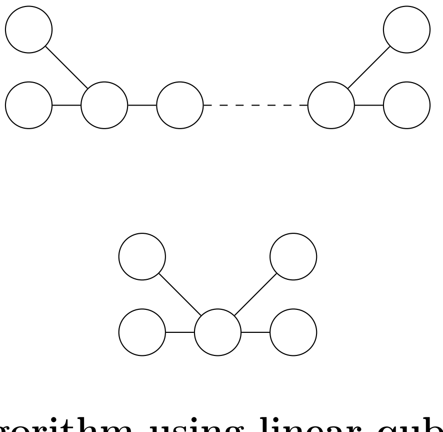
For any graph, we are looking for a procedure that minimises the number of mergers (we will assume that a linear chain or star has a "null cost").

In the following, we have to work on graphs of photons - the graphs that we want to build for the quantum computations. We will represent the vertices of a photons graph by big empty circles.

In a photons graph (see Fig. 3.1):

- A simple edge between two photons means that they have been directly entangled.
- A dashed line between two photons expresses the fusion between these two photons.

Figure 3.1: Example of two equivalent photons graphs



3.1 A first "naive" algorithm using linear qubits chains

3.1.1 Principle

We were first interested in solving a simplified problem, in which our basic "bricks" were linear chains of any length.

The graph, provided in the form of an adjacency list, is traversed in width or depth, starting from any vertex, associated with a chain of length 1.

- If the current vertex has no unprocessed neighbour, the next vertex in the list of discovered but unprocessed vertices is processed.

Figure 3.2: Addition of an extra neighbour in the form of a 3-qubit chain

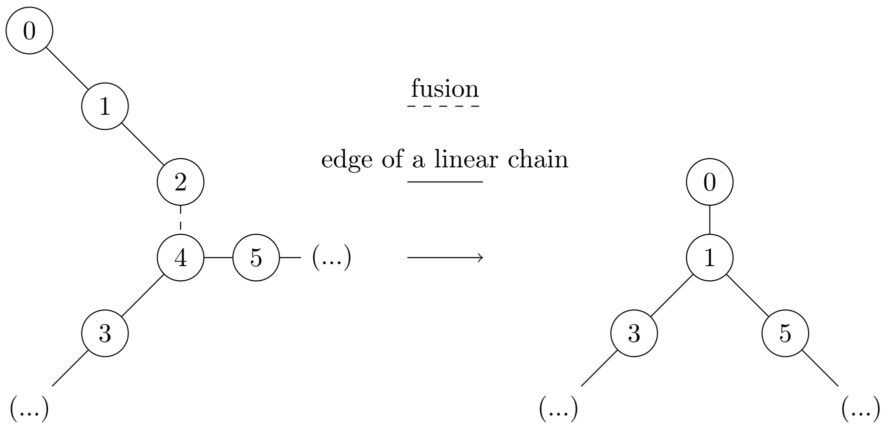
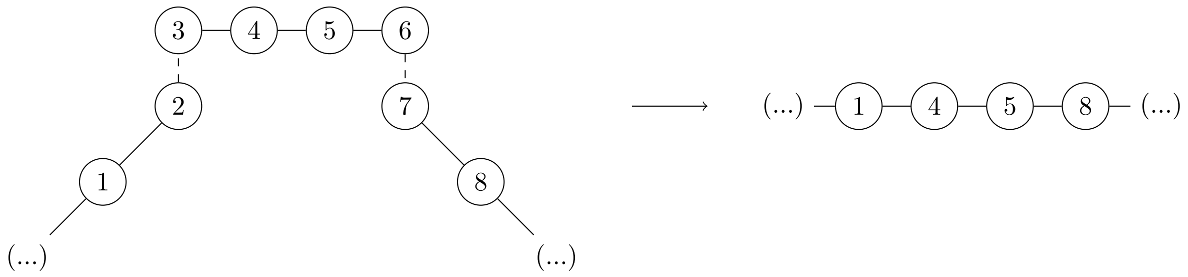


Figure 3.3: Creation of an edge between two already explored vertices by means of a 4-qubit chain



- If the current vertex has only one undiscovered neighbour, it is added to the end of the chain to which it belongs.
- If the current vertex has several undiscovered neighbours, one is added to the end of the chain to which it belongs, and a chain of 3 qubits is created for each of the other unprocessed neighbours, merged with the appropriate vertex (see Fig. 3.2).

Furthermore:

- All undiscovered neighbours are added to the list of discovered vertices.
- For each of the neighbours of the current vertex that has been discovered, we create a chain of 4 qubits to create the desired connection (see Fig. 3.3).
- The current vertex is finally added to the list of processed vertices.

The algorithm stops when all the vertices have been processed.
Note that a table is constantly updated associating each discovered or processed qubit with its chain number and its place in the chain, so that no qubit is merged twice (which is of course physically impossible !).
The algorithm returns:

- a list of integers representing the lengths of the linear chains necessary for the realization of the cluster state
- a list of couples $(n°\_chain\_1, place\_chain\_1; n°\_chain\_2, place\_chain\_2)$ representing the fusion operations.

3.1.2 Improvement of the result

The result obtained, although correct, is still far from optimal. It can be brought closer by noting that, in some configurations, mergers are supposed to create an edge between nodes located at the ends of two distinct chains, which is obviously superfluous, the basic structures ("bricks") here being linear chains. A post-processing algorithm allows us to significantly reduce the number of merges, by linking the chains together in this situation (see Fig. 3.4).

Figure 3.4: Reduction of the number of mergers after application of the post-processing algorithm

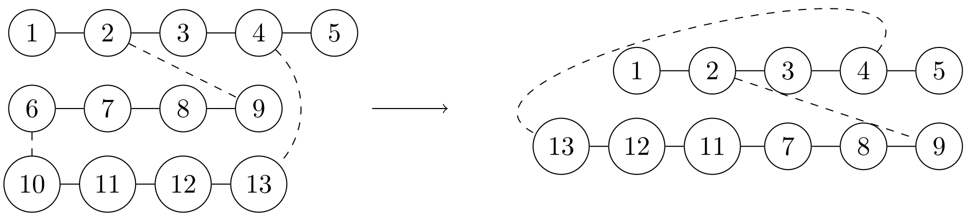
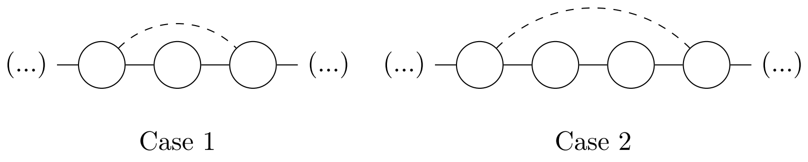


Figure 3.5: Prohibited situations resulting from the application of (R') in two particular configurations.



3.1.3 Proof

In the case of our algorithm, the merger rule stated here, (R), reduces to the simpler rule (R'): when merging (a) and (b), all neighbours of (a) are connected to all neighbours of (b), and (a) and (b) (and adjacent edges) are removed from the graph.
It is thus clear that at each stage of the graph traversal, the graph already constructed corresponds to the subgraph comprising the set of processed and discovered vertices, and the set of edges starting from the processed vertices: the algorithm before post-processing is correct (P1).
Finally, we seek to show that the graph constructed after post-processing is satisfactory (P2). This is true if the order of the mergers returned does not affect the final graph (P3). We notice that, if we apply (R') to the result, the order of the mergers does not matter (P4). However, if we apply (P4) to cases 1 and 2 shown in Fig. 3.5, we obtain physically impossible situations: in the first case, an edge links a vertex to itself, and in the second, two vertices are linked by two edges. Thus, if either (case 1) or (case 2) occurs, the resulting graph is incorrect. However, (P4) and (P1) imply that the graph obtained by applying (R') in any order is correct. Thus, (case 1) and (case 2) do not occur, so (R) reduces to (R') regardless of the order of the mergers. Finally, from (P4), (P5) is true, from which we deduce (P2).

3.1.4 Test

We have tested our algorithm on randomly generated graphs, using the algorithm presented below. The procedure returned by the main algorithm was then applied and the result compared to the original graph. This allowed us to confirm the correctness of our program, which you can see in full in this repository: https://gitlab-student.centralesupelec.fr/hector.blondel/qoptique.

3.1.4.1 Random graph generation

To test our algorithm we needed a way to provide some graphs. To do so we chose to generate random graphs using a function from the Python library *igraph*, which need as input a list of node degrees and output a ranndom graph with nodes of the corresponding degrees. This algorithm uses a Monte-Carlo method to generate the random graph. For more information check [7]

3.1.4.2 Reconstitution of the original graph

We have written an algorithm that reconstitutes a graph from the decomposition.

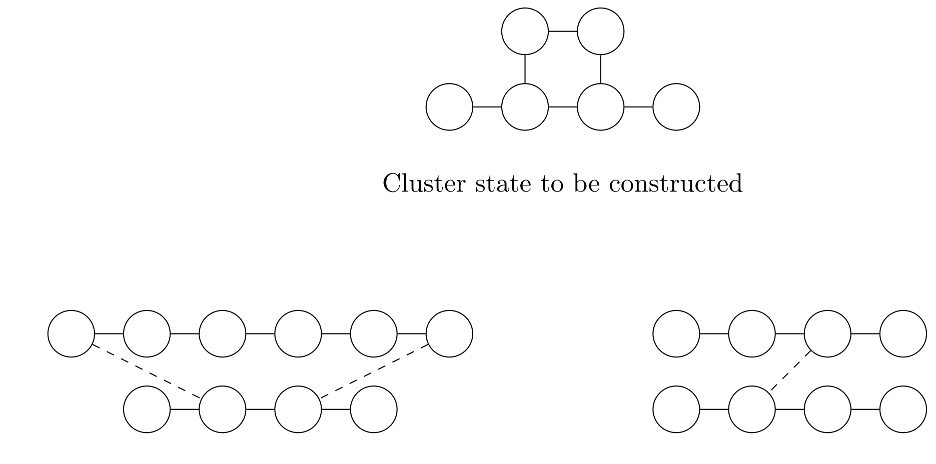
It takes as input an adjacency dictionary representing the graph, and the list of fusions, it goes through the list of fusions, studying the qubits of each fusion, looking for their neighbors, and performing the fusion.

```
def reconstitution(dictionnaire_dadjacence, fusions):
    for i in range(len(fusions)):
        qubit1, qubit2 = fusions[i]
        voisins_de_qubit1,voisins_de_qubit2 = dictionnaire_dadjacence[qubit1],dictionnaire_
        for voisin in voisins_de_qubit1:
            dictionnaire_dadjacence[voisin] += voisins_de_qubit2
            dictionnaire_dadjacence[voisin].remove(qubit1)
        for voisin in voisins_de_qubit2:
            dictionnaire_dadjacence[voisin] += voisins_de_qubit1
            dictionnaire_dadjacence[voisin].remove(qubit2)
        del dictionnaire_dadjacence[qubit1]
        del dictionnaire_dadjacence[qubit2]
    return dictionnaire_dadjacence
```

3.1.5 Non-optimality

This algorithm, although correct, is sometimes far from optimal, notably because it substitutes a sub-rule (R') for the fusion rule (R) (see section 3.1.3). An exemple can be seen in Fig. 3.6. Therefore, we subsequently looked for an efficient method to explore the set of correct solutions, allowing to better take into account the possibilities offered by the type II fusion.

Figure 3.6: Situation in which the naive algorithm is not optimal


Cluster state to be constructed
Procedure returned by the naive algorithm
Optimal procedure

3.2 Towards an effective solution

3.2.1 A first fusion graph formalism, using qubits linear chains

In order to optimize the solution provided by the algorithm, we have chosen to focus on the research of transformation rules between different solutions which lead to the same graph.

Our approach is based on two facts:

- Every single graph can be expressed, thanks to our algorithm, as the result of fusions between chains of photons.
- Every single photon chain containing more than three photons can actually be expressed as the result of the fusion of chains of three photons. Furthermore, 3 is the minimum number of photons in a chain for which the property holds: the fusion between two 2-photons chains only leads to a new 2-photons chain.

Hence it is possible to express a solution as a set of fusions between chains of three photons. We will represent a solution as a graph in which a vertex represents a chain of three photons and an edge represents a fusion between two 3-photons chains. In the following we will call it a fusion graph. In order to distinguish it from photons graphs the vertices of a fusion graph will be represented by smaller filled circles.

In a fusion graph (see Fig. 3.7 and Fig. 3.8):

- A simple edge represent the fusion between the photons at the extremities of two chains.
- An arrow from a vertex A to a vertex B expresses the fusion between one extremity of the chain A and the center of the chain B.
- A double arrow between two vertices expresses the fusion between the centers of the two chains.

Figure 3.7: Example of a fusion graph, the corresponding photons graph and the resulting photons graph (after the fusions)

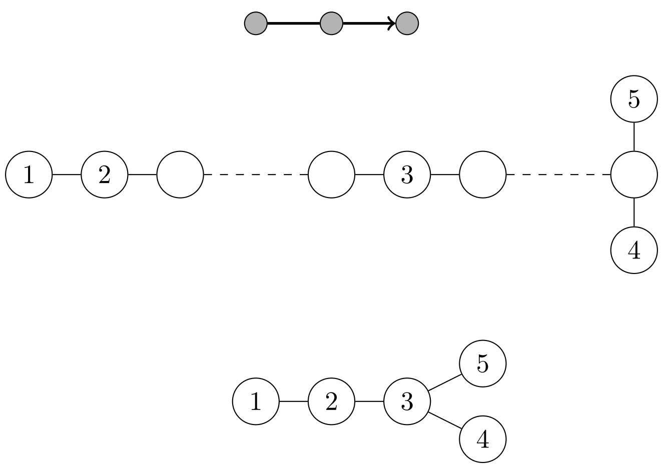
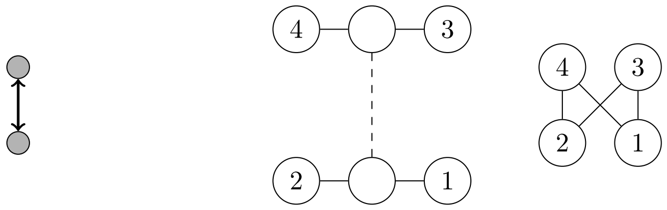


Figure 3.8: Another example, involving the double arrow

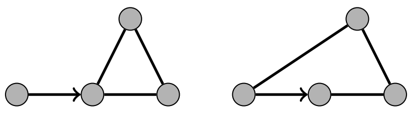


3.2.2 Equivalence and graph conversion

3.2.2.1 Motivation example of our approach

The two fusion graphs of Fig. 3.9 are equivalent to the extent that the resulting photons graph they encode is the same: see Fig. 3.10 Looking to the left-hand graph, it seems that the optimal fusion pattern is to process two fusions: a first one between the two extremities of a chain of five photons, in order to produce a cycle of three photons, and a second one between one of the photons of the cycle and the extremity of a chain of three photons. But looking to the equivalent right-hand graph, it appears that only one fusion is required: between two photons of a chain of 6 photons. These two patterns are displayed in Fig. 3.11 Since the fusion operation is not deterministic and has a probability of failure, it appears that, assuming that producing a single 6-photons chain is not more difficult than producing a 3-photons chain and a 5-photons chain, the right-hand pattern has the higher success probability and is thus the optimal fusion pattern.

Figure 3.9: Two equivalent fusion graphs



This example shows that performing simple transformations on a fusion graph can highlight a more efficient fusion pattern.

3.2.2.2 Even more symbols

In order to describe general conversion rules on fusion graphs, we have to isolate elementary patterns from the rest of the graph and then describe how to modify them.

On photons graphs, we will track some photons by assigning them an id, as we have done in Fig. 3.7. This will be useful to get track of the photons that are potentially involved in a fusion out of the pattern we are describing.

When the photon is an extremity of its 3-photons chain, the corresponding potential fusion can be on the fusion graph either a simple edge or an outgoing arrow. We will represent on the the position of the tracked photon, and thus the potential fusion, by a bar separating the node and the id of the tracked photon (see Fig.3.12).

Figure 3.10: Resulting photons graph

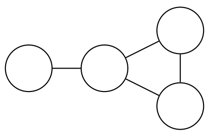
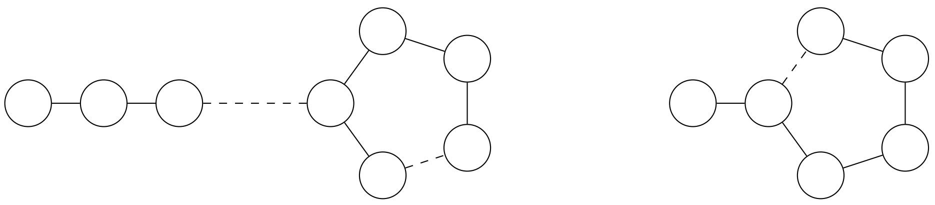


Figure 3.11: Two different patterns leading to the same result



3.2.2.3 Simple rules

The first rule concerns the symmetry of n-pointed stars. The figure 3.13 shows an example of pattern leading in the photons graph to a three-pointed star. In this figure, one can notice that the roles of the photons 1,2 and 3 are perfectly symmetric. Thus any permutation on the possible connections 1,2 and 3 of the fusion graph lead to the same photons graph.

Furthermore, one can notice that an arrow pointing on a three-pointed star lead to a four-pointed star as shown in Fig. 3.14. Once again, the roles of the four extremities of the star are perfectly symmetric and any permutation of the four potential fusions on the fusion graph leads to the same photons graph.

More generally, any linear series of n arrows in the fusion graph encodes a (n+2)-pointed star of which the roles of the n+2 extremities are symmetric.

The Fig. 3.15 shows two equivalent fusion graphs leading to a five-pointed star. The right-handed graph can be obtained from the left-handed one by exchanging the fusions a and 1 (using the symmetry of the 3-pointed star), then exchanging the fusions 3 and 5. This first example of application of the symmetry rule shows that in an n-pointed star, the sense of the arrows doesn't matter and can also be modified.

Note: This rule enables to switch between the two graphs of Fig. 3.9.

The second rule concerns the fusion between the centers of two 3-photons chains. The Fig. 3.16 demonstrates step by step the equivalence between a double arrow between two nodes and a cycle of four nodes. The top fusion graph (a) directly encodes the left-handed photons graph (b). From left to right (b to e), the photons graphs are derived by applying simple transformations. Finally the bottom fusion graph (f) directly encodes the right-handed photons graph (e). Each transformation conserves the equivalence of the resulting photons graph, the two fusion graphs (a and f) are therefore equivalent.

3.2.2.4 Application example

The figures 29 and 28 in the appendices show an example of the use of the rules just defined. The target is to obtain the photons graph 29a. The simplest way to obtain it is to set each photon of the target graph as the center of a star, and get the links between them by applying fusions on the branches of the stars. This leads to the graph 29b. By directly expressing each star as a sequence of arrows in the fusion graph, one gets the graph 29c. The graph 29d is derived from 29c by applying the rule of symmetry of stars on each of the three-pointed and four-pointed stars. Two four-cycles are then exhibited, which can be simplified by applying the rule of equivalence with double arrows, leading to the graph 29e. This last fusion graph directly encodes the photons graph 28a. By operating some of the fusions of this graph, one can derive the final photons graph 28b which exhibits only tree fusions between two photons chains.

Finally, the use of conversion rules just enabled to simplify a photons graph (29b) involving 33 photons and 12 fusions into a simpler photons graph (28b) involving only 15 photons and 3 fusions. As the production of entangled photons and especially the fusion operation introduce a probability of failure, this last graph clearly appears as more optimal than the first.

3.2.3 A second formalism, using qubits stars

Most of the procedures we describe in this section are implemented in the file *star/star_graph.py* of our code.

As stars regularly appear during the optimisation phase, we decided to extend our formalism to the description of stars of photons. An n-pointed star is a photon, which we will call the center, surrounded by n entangled

Figure 3.12: Potential fusions in fusion graph and the corresponding photons graph

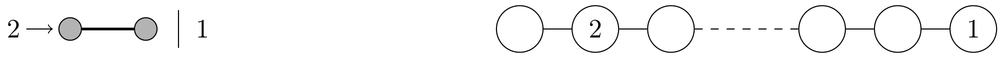
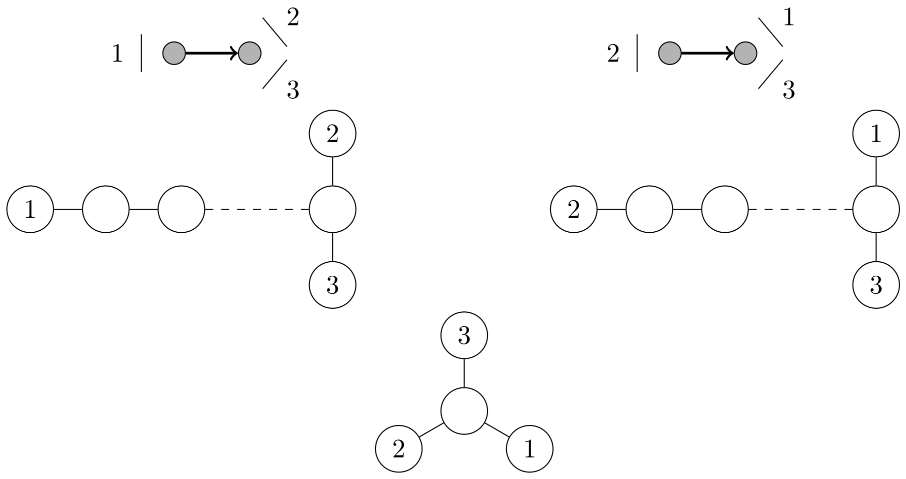


Figure 3.13: Equivalent fusion graphs leading to the same three-pointed star

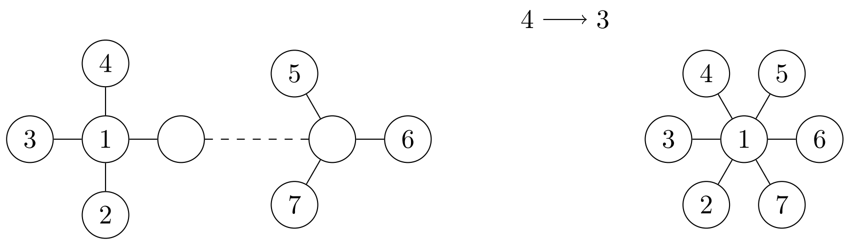


photons, the extremities. The previous figures 3.13 and 3.14 give examples of stars. As a chain of three photons is a 2-pointed star, this formalism is an extension of the previous one.

We will then operate on graphs involving only stars, which we will call stars graphs, in which an n-pointed star is simply represented by a floating n. The previous types of fusion still hold:

- The fusion between the extremities of two stars, denoted by a simple edge. We will call it a peripheral fusion. An n-pointed star must have exactly n peripheral fusions with n different stars. In order to simplify the figures we might draw less than n peripheral fusions: the remaining fusions are then free and can be set with any other star.
- The fusion between the centers of two stars, denoted by a double arrow. We will call it a central fusion.
- The fusion between the center of a star and an extremity of another star, denoted by a simple arrow, still has a sense but might be useless. Indeed, such a fusion between an extremity of an n-pointed star a and the center of an m-pointed star b only results in a (n+m-1)-pointed star: the center of b and the extremity of a have been consumed by the fusion, and all the m extremities of b have become entangled with the center of a in addition to the n-1 remaining extremities of a. The Fig. 3.17 gives an example of such a fusion.

Figure 3.17: Merge of a 4-pointed star and a 3-pointed star



One could notice that an peripheral fusion with a 1-pointed star has no effect on the final photons graph as the center of the 1-pointed star only substitutes to the consumed extremity of the other star. In the same way, an central fusion with a 1-pointed star has no effect on the final photons graph. Therefore, 1-pointed stars are useful for the calculations but have to be suppressed at the end of the optimisation.

3.2.4 Generation of an elementary star graph using peripheral fusions

We can easily generate a star-graph state which can lead to a given graph state.

- In the implementation, we authorize 1-branch photon chains. These chains are composed of two photons, and in a representation, we can make them disappear when they are connected to another star.

Figure 3.14: A four-pointed star in the fusion graph an the corresponding photons graphs

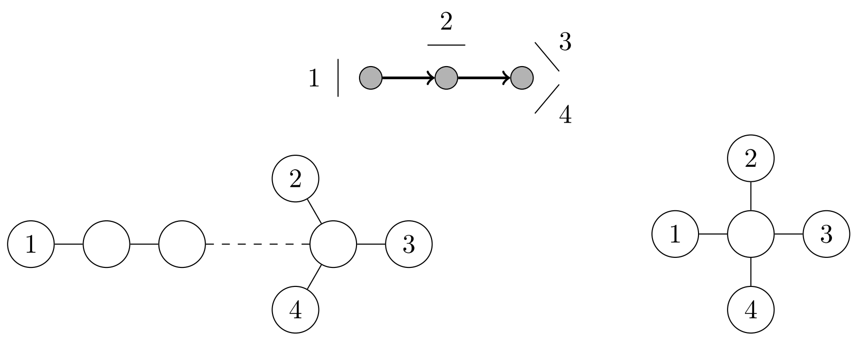
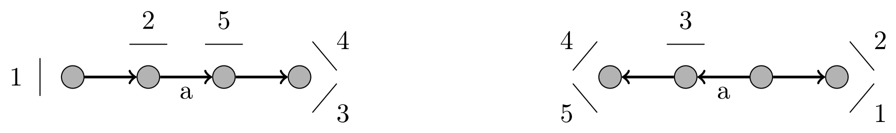
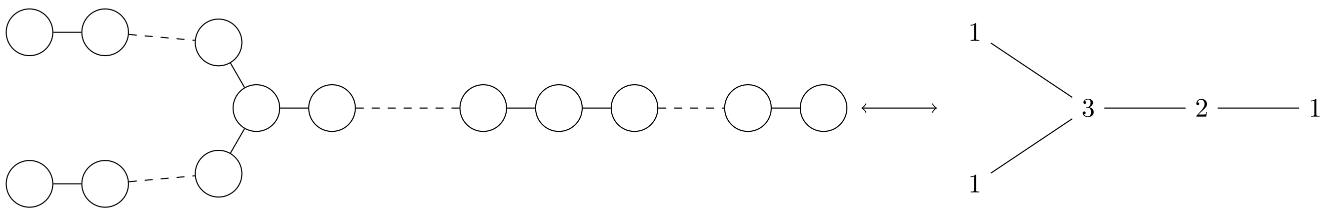


Figure 3.15: Example of equivalent 5-pointed stars



- Given the objective graph state, we generate for each photon a star with the number of branches being the number of neighbours.

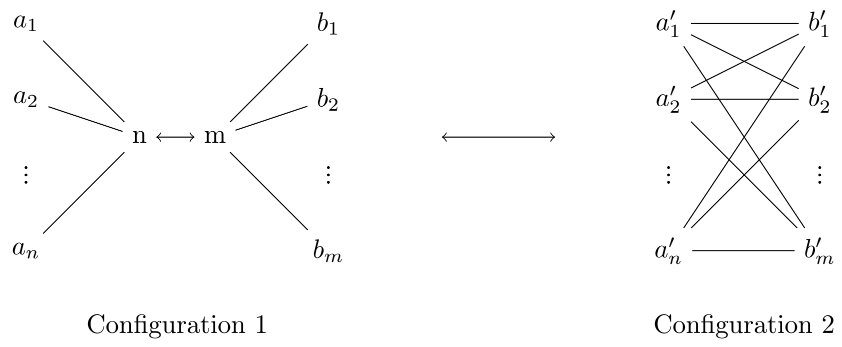
In the *Graph* class of out implementation, we don't need to perform any calculation during the initialisation. Indeed, the tag of each node (i.e. the number of branchs of the corresponding star) is always the number of neighboors, and is recalculated each time the program need to access it.



3.2.5 General transformation rule

This simpler formulation allows us to consider a generic rule that can represent all possible transformations. We only have two types of star fusion, one of which, central fusion, cannot be performed twice on the same star (because its central qubit would be involved in two fusion gates, which is physically impossible). Any transformation involving a central fusion can therefore be represented by Fig. 3.18 (we will call it a "central transformation"). Note that m and n are integers greater than 1 (see section 3.2.3).

Figure 3.18: General transformation rule using a central fusion ($a'_i = a_i - 1 + m$ and $b'_j = b_j - 1 + n$)

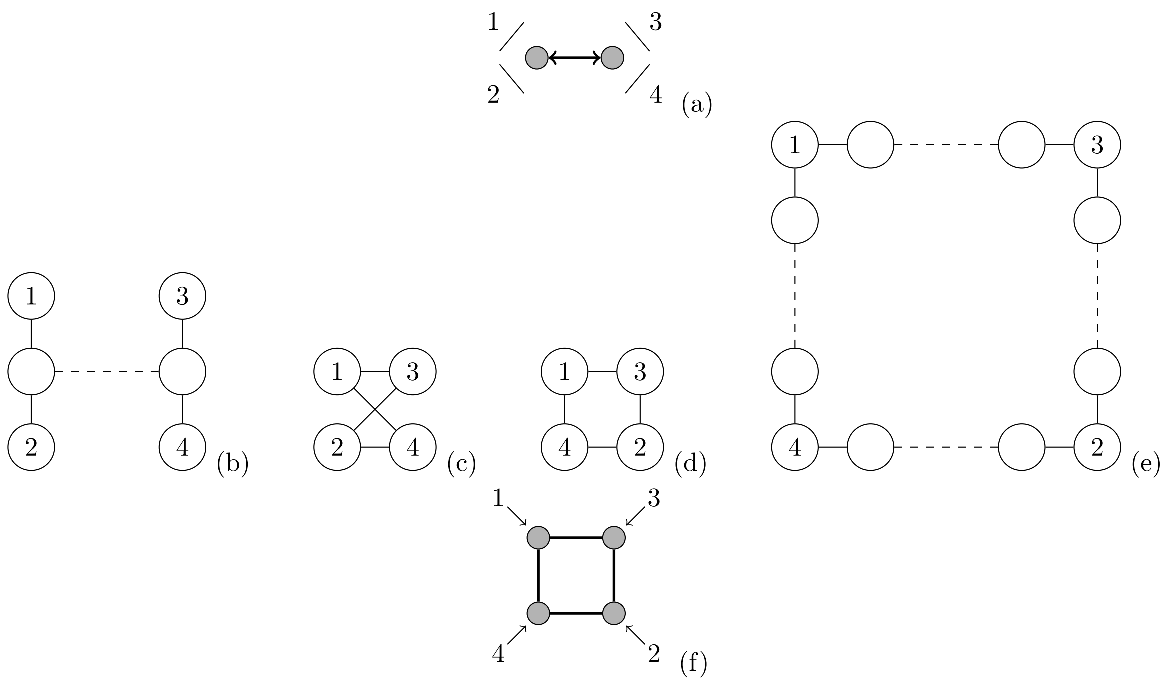


Configuration 1 Configuration 2

We have seen previously that each cluster state can be realised using peripheral fusions between stars.

We also notice that, in this situation, each node of the star graph is associated, in the final graph, with a qubit entangled with a number of neighbours equal to its label, a remark that can be extended to each of its neighbours. There is thus an obvious bijection between the set of qubit cluster states and the set of star graphs

Figure 3.16: Demonstration of the rule of equivalence between double arrows and four-cycles



involving only peripheral mergers: the representation found in section **??** is unique.
It follows that any transformation of a star graph takes the form of a succession of central transformations: the situation considered in Fig. 3.18 is in fact the general case.

Mergers involving one-branch stars, while relevant in a theoretical framework, are unnecessary in practice, and can be replaced by unmerged branches of primitive stars. That way, in configuration 1, we finally perform $m + n + 1 - m' - n'$ fusion operation (m' and n' being the number of one-branch stars linked to m and n respectively in configuration 1), while in configuration 2 we perform $n \times m$ fusions.
Given a pattern associated with one of the configurations 1 and 2, we are thus able to simply determine whether a central transformation leads to a decrease or an increase in the total number of mergers, which opens the way to a first greedy algorithm.

3.2.6 Search for complete bipartite subgraphs (CBS)

In the following, we will note CBS for "Complete Bipartite Subgraph".

In order to perform interesting simplifications, we need to detect the patterns shown in figure 3.18. The central fusion (configuration 1) can easily be detected since it is directly encoded in the data structure, but the configuration 2 is more complex to detect: it is a complete bipartite subgraph.

A graph is said to be *bipartite* if one can classify its vertices into two disjoint sets, called the parts of the graph, such that two vertices of a single set cannot be neighbours. It is a complete bipartite graph if all the vertices are neighbours with all the vertices of the other part.

Let us denote by $N(v)$ the set of all the neighbours of the vertex v.

For example in configuration 2 (see Fig. 3.18), the two parts are $A = \{a'_i | i \in [\![1; n]\!]\}$ and $B = \{b'_i | i \in [\![1; m]\!]\}$. The graph is bipartite since $\forall i, j \in [\![1; n]\!], a'_j \notin N(a'_i)$ and $\forall i, j \in [\![1; m]\!], b'_j \notin N(b'_i)$. Furthermore, the bipartite graph is complete since $\forall i \in [\![1; n]\!], B = N(a'_i)$ and $\forall i \in [\![1; m]\!], A = N(b'_i)$.

In the graph we want to simplify, edges can exist between two vertices of a same part of the subgraph we are inspecting. The only important thing is that by deleting some edges we can extract a complete bipartite subgraph.

3.2.6.1 Our first heuristics

In order to be efficient, the algorithm needs to find relatively quickly complete bipartite subgraphs. The larger the graph is, the more interesting it is to convert it into a central fusion. Since the detection and the conversion of a cbs have to be iterated, the algorithm doesn't need to find the best cbs first nor to find all the possible cbs. The only requirement is to find at least an interesting cbs if there exists at least one.

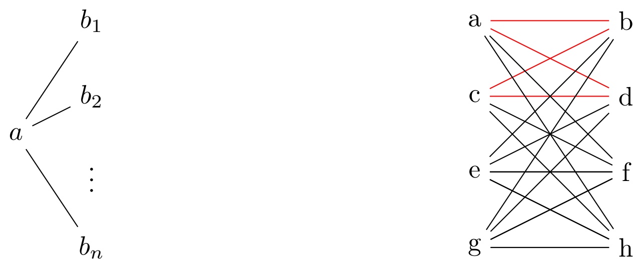
The idea of the heuristics is to identify a smal cbs that is easy to find, and extend it to a cbs that is as large as possible. The heuristic iterates the addition of vertices to the detected cbs, one by one, insuring that at any iteration the subgraph is still a cbs.

Important remark: in an interesting cbs, the two parts contain at least two vertices. Indeed, if one of the parts contains only one vertex, then the subgraph is simply a star surrounded by its neighbours, as shown in Fig. 3.19a. This cannot be simplified and shall then be ignored by the algorithm. Therefore interesting subgraphs contain at least two vertices per part. As shown in Fig. 3.19b, it implies that the cbs contains at leat one 4-cycle : a cylcle of four vertices. As a 4-cycle is a complete bipartite subgraph, it is the littlest cbs that is interesting to detect. The firts step of the algorithm is then to detect a 4-cylce in the graph. Since this problem is quite common and have probably already been solved efficiently many times, we haven't tried to develop our own 4-cylce detection algorithm.

Figure 3.19: Examples of particular graphs

(a) Complete bipartite subgraph which is actually a star

(b) Large enough complete bipartite subgraph



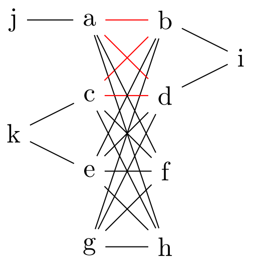
The steps of the heuristics are the following :

1. Find a 4-cycle.
2. Build the two parts of the cbs. Let's call them V_1 and V_2.
3. Enumerate the common neighbours of the vertices of each part, and subtract the vertices that have already been added to the cbs. Let us call these sets $C_1 = \bigcap_{v \in V_1} N(v) \backslash (V_1 \cup V_2)$ and $C_2 = \bigcap_{v \in V_2} N(v) \backslash (V_1 \cup V_2)$
4. Choose a vertex to add in the cbs. If possible, equilibrate the two parts by adding a vertex to the smaller one. This vertex has to be chosen among the common neighbours of the other part and have a maximal number of neighbours among the common neighbours of its part. For example, if we want to add a vertex to V_1, we will choose $v^* \in C_2$ such that $|N(v^*) \cap C_1|$ is maximal. Then add v^* to V_1. By choosing a vertex in C_2 we ensure that it will be a neighbour of each vertex of V_2 : the subgraph will still be complete bipartite. By choosing it such that $|N(v^*) \cap C_1|$ is maximal, we maximize the number of potential vertices we will be able to add in V_2. Here lies the heuristics.
5. Iterate from step 3 until $C_1 = C_2 = \emptyset$: no more vertex can be added.

Let us give an example of the execution of the heuristics. First, detect a 4-cycle in the graph and build the two initial parts of the cnbs. Then, compute the common neighbours of each part. In Fig. 3.20, we get :

$$
\begin{aligned}
V_1 &= \{a, c\} \\
V_2 &= \{b, d\} \\
C_1 &= \{ f, h \} \\
C_2 &= \{ e, g, i \}
\end{aligned}
$$

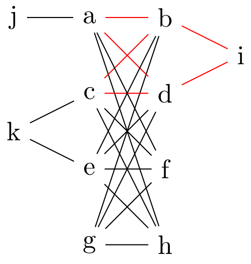
Figure 3.20: Initial graph



V_1	$a \to f, h, j$
	$c \to f, h, k$
V_2	$b \to e, g, i$
	$d \to e, g, i$

If we added the vertex i to the part V_1, we would get the graph in Fig. 3.21 in which $C_1 = \emptyset$. It would be impossible then to add f and h and the algorithm would not detect the massive central cbs that is visible on Fig. 3.25. The aim of the selection rule is then to avoid this situation to happen.

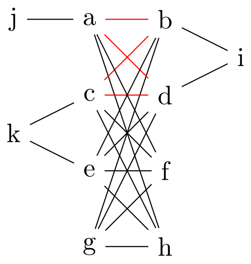
Figure 3.21: Graph with wrong vertex added



V_1	$a \to f, h$
	$c \to f, h$
	$i \to \emptyset$
V_2	$b \to e, g$
	$d \to e, g$

Therefore, for each of the common neighbours of a part, we count the number of neighbours among the common neighbours of the other part. We can see Fig. 3.22 that for example f has two neighbours in C_2 : e and g. As we noticed previously, i has no neighbour in C_1.

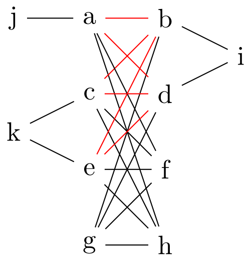
Figure 3.22: Initial graph, selection criteria



V_1	$a \to f_2, h_2, j_\emptyset$
	$c \to f_2, h_2, k_\emptyset$
V_2	$b \to e_2, g_2, i_0$
	$d \to e_2, g_2, i_0$

If we want to add a vertex to V_1 its counter has to be maximal. Hence it cannot be i. Let us choose e. We obtain the graph Fig. 3.23.

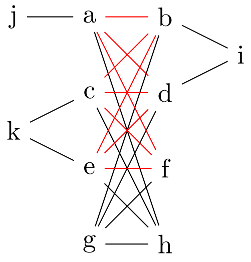
Figure 3.23: Addition of the first vertex



V_1	$a \to f_1, h_1, j_\emptyset$
	$c \to f_1, h_1, k_\emptyset$
	$e \to f_1, h_1, k_\emptyset$
V_2	$b \to g_2, i_0$
	$d \to g_2, i_0$

In order to balance the parts, let us add a vetex to V_2. It could be either f or h. Let us choose f. We get the graph Fig. 3.24. One can note that i does not belong to C_2 anymore and thus will not be able to be added to the cbs : the heurisctics worked !

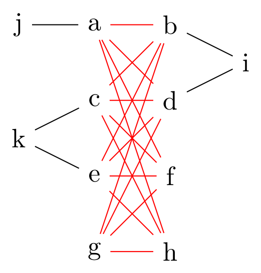
Figure 3.24: Addition of the second vertex



V_1	$a \to h_1, j_\emptyset$
	$c \to h_1, k_\emptyset$
	$e \to h_1, k_\emptyset$
V_2	$b \to g_1, i_\emptyset$
	$d \to g_1, i_\emptyset$
	$f \to g_1$

The next vertices to be added are h and g, leading to the final graph Fig. 3.25. No more vertex can be added : the heuristics ends.

Figure 3.25: Final graph



V_1	$a \to j_\emptyset$
	$c \to k_\emptyset$
	$e \to k_\emptyset$
	$g \to \emptyset$
V_2	$b \to i_\emptyset$
	$d \to i_\emptyset$
	$f \to \emptyset$
	$h \to \emptyset$

3.2.7 Our optimization algorithm

On the basis of elements exposed previously, here is an algorithm we have proposed for solving the initial problem on stars :

1. Start from a configuration given by the (simple) procedure described in 3.2.4.
2. For each configuration, we detect bipartite subgraphs (for that purpose we used in out code the heuristics of [6], but we could have used the algorithm described in 3.2.6) and double-arrows. The detection of bipartite subgraphs is perfomed in the method *find_ nonoverlapping_ bcss*.
3. For each CBS or subgraph is associated a transformation and a corresponding decreasing in the number of fusion. From configuration 1 to configuration 2, the decreasing is $mn - m - n - 1 + m' + n'$ and in the other way it is $m + n + 1 - m' - n' - mn$. The transfomation_step method is in charge of doing such comparisons.
4. Choose one of the transformations, apply them by the rule described in 3.2.5, and return to step 2.

The procedure we just described is an optimization algorithm, and it can use different kind of choices for the transformation to perform :

- Greedy : Always take the transformation which leads to the least-fusion configuration
- Simulated annealing (randomized algorithm)

In our implementation, we used the greedy approach as a basic algorithm.

3.2.8 Results

The following graph, we have represented the ratio number of fusions at the end over the number of fusions at the beginning of our algorithm.

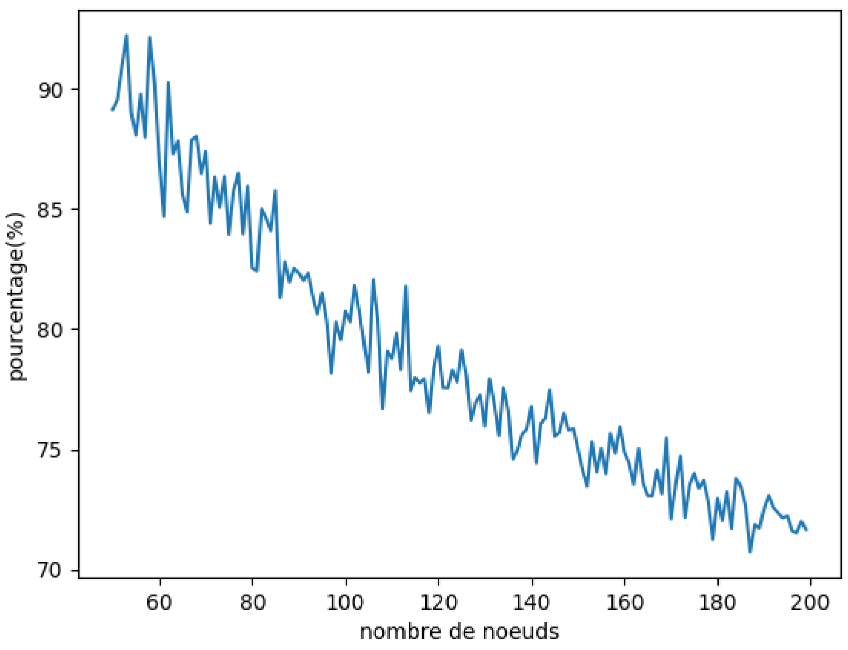
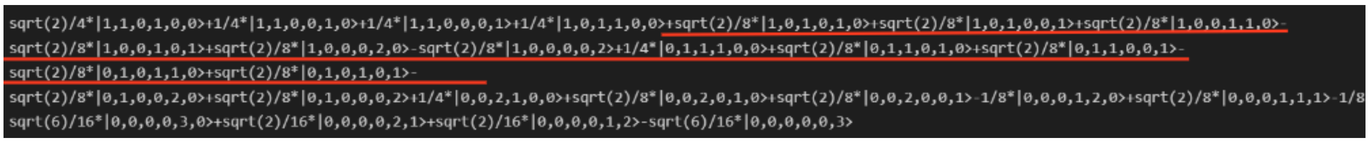


Figure 3.26: Rate of won fusions compared to the naive graph

Conclusion

In the course of this project, we have therefore succeeded in determining and simulating procedures that generate entangled states. Once obtained, these states can be the subject of a series of measurements leading to the result of a specific quantum calculation. This method, known as the measurement-based quantum computation, enables basic operations to be carried out in which any quantum gate can be decomposed: it is universal, and could therefore enable photonic quantum computers to emerge.

Appendices

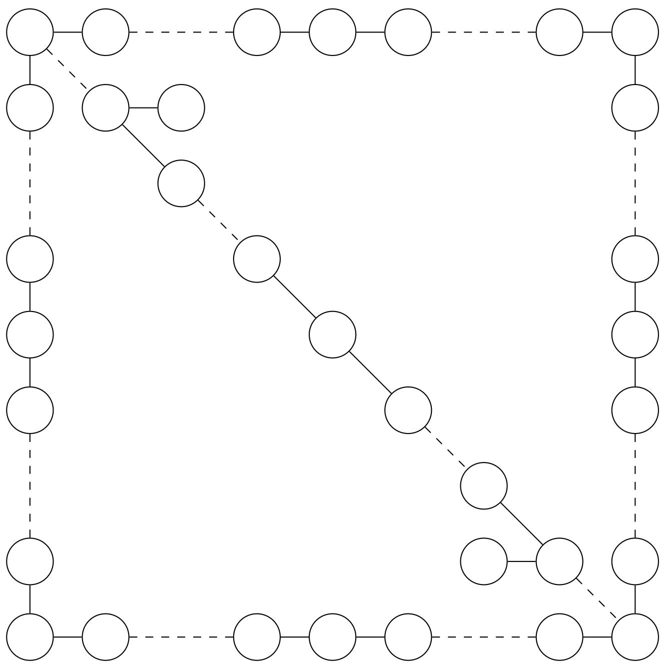


```
sqrt(2)/4*|1,1,0,1,0,0>+1/4*|1,1,0,0,1,0>+1/4*|1,1,0,0,0,1>+1/4*|1,0,1,1,0,0>+sqrt(2)/8*|1,0,1,0,1,0>+sqrt(2)/8*|1,0,1,0,0,1>+sqrt(2)/8*|1,0,0,1,1,0>-
sqrt(2)/8*|1,0,0,1,0,1>+sqrt(2)/8*|1,0,0,0,2,0>-sqrt(2)/8*|1,0,0,0,0,2>+1/4*|0,1,1,1,0,0>+sqrt(2)/8*|0,1,1,0,1,0>+sqrt(2)/8*|0,1,1,0,0,1>-
sqrt(2)/8*|0,1,0,1,1,0>+sqrt(2)/8*|0,1,0,1,0,1>-
sqrt(2)/8*|0,1,0,0,2,0>+sqrt(2)/8*|0,1,0,0,0,2>+1/4*|0,0,2,1,0,0>+sqrt(2)/8*|0,0,2,0,1,0>+sqrt(2)/8*|0,0,2,0,0,1>-1/8*|0,0,0,1,2,0>+sqrt(2)/8*|0,0,0,1,1,1>-1/8
sqrt(6)/16*|0,0,0,0,3,0>+sqrt(2)/16*|0,0,0,0,2,1>+sqrt(2)/16*|0,0,0,0,1,2>-sqrt(6)/16*|0,0,0,0,0,3>
```

Figure 27: Output state of circuit of figure 2.5, linear cluster state of length 3 is underlined

Figure 28: Resulting photons graphs

(a) encoded photons graph



(b) final photons graph

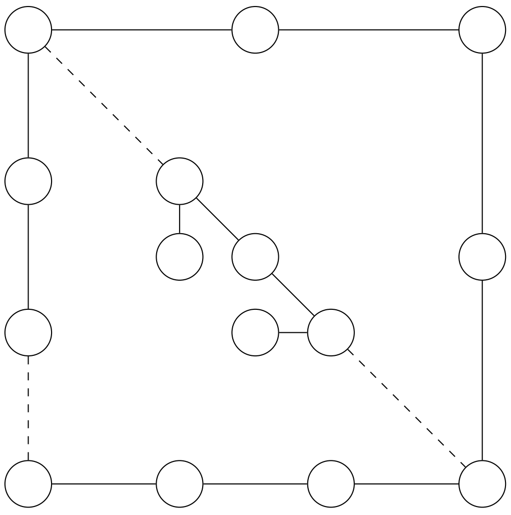
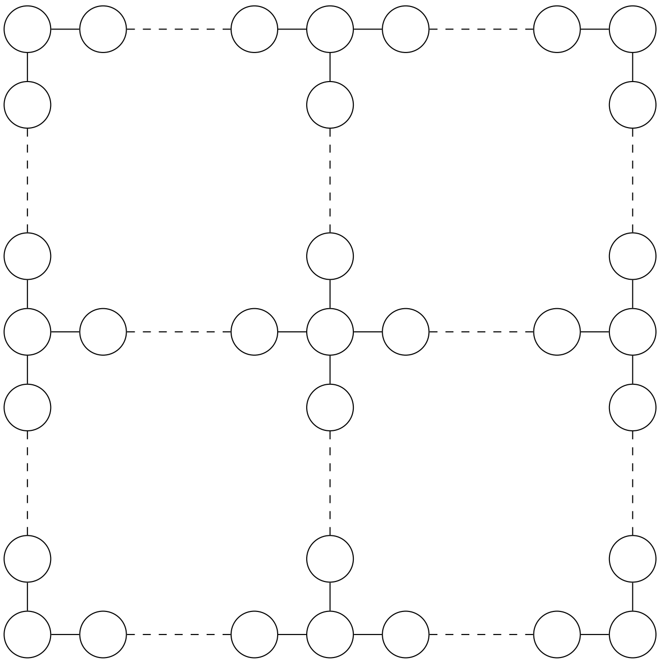
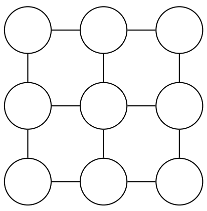


Figure 29: Application of the rules on photons chains

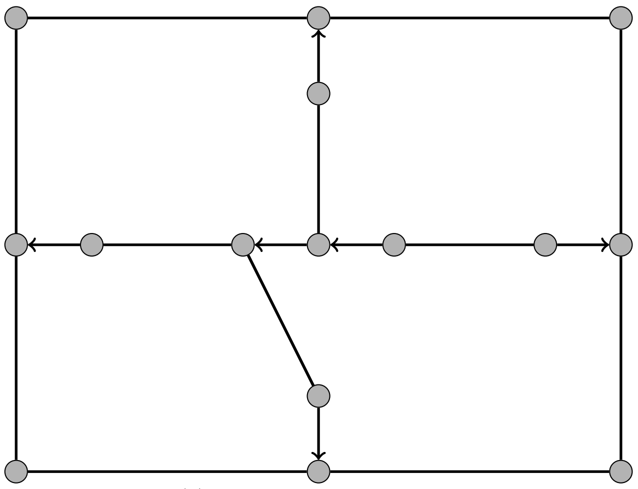
(b) first photons graph



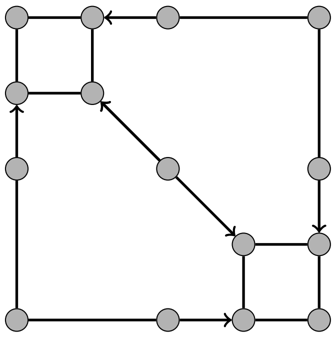
(a) target



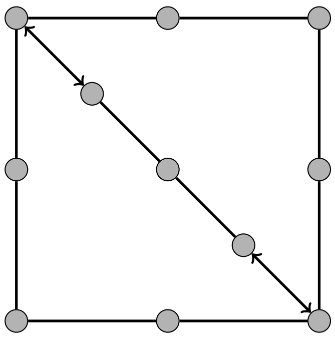
(c) first fusion graph



(d) reshaped fusion graph



(e) simplified fusion graph



Bibliography

[1] Mark Fox. *Quantum Optics : An Introduction.* Oxford University Press, Oxford, 2006.

[2] Mercedes Gimeno-Segovia. Towards linear quantum computing, November 2015.

[3] Nicolas Heurtel, Andreas Fyrillas, Grégoire de Gliniasty, Raphaël Le Bihan, Sébastien Malherbe, Marceau Pailhas, Eric Bertasi, Boris Bourdoncle, Pierre-Emmanuel Emeriau, Rawad Mezher, Luka Music, Nadia Belabas, Benoît Valiron, Pascale Senellart, Shane Mansfield, and Jean Senellart. Perceval: A software platform for discrete variable photonic quantum computing. *Quantum*, 7:931, feb 2023.

[4] Jiri Vala Javier Osca. Implementation of photon partial distinguishability in a quantum optical circuit simulation. 2023.

[5] E. Knill. Quantum gates using linear optics and postselection. *Physical review A*, 66(052306), 2002.

[6] Hyunseok Jeong Seok-Hyung Lee. Graph-theoretical optimization of fusion-based graph state generation. *Quantum physics*, 2023.

[7] Fabien Viger and Matthieu Latapy. Efficient and simple generation of random simple connected graphs with prescribed degree sequence. *Journal of Complex Networks*, 4(1):15–37, 2016.