

ÉCOLE CENTRALE DE LILLE - UNIVERSITÉ DE LILLE



EQUIPE PROJET INRIA PARADYSE LILLE - LAB. P.PAINLEVE CNRS

Vers l'avantage quantique ?

Auteur :

Victor Niaussat

Encadrant :

Stephan De Bièvre

8 mars 2023

Table des matières

1	Introduction	1
1.1	Prélude et contexte de travail	1
1.2	Problème étudié	1
2	Boson Random Sampling	4
2.1	Formalisme mathématiques de l'optique quantique	4
2.1.1	Décrire un état de la physique	4
2.1.2	Opérateur de création et d'annihilation pour les bosons	6
2.1.3	Évolution dans le temps d'un état ou d'un opérateur	7
2.1.4	Modèle d'évolution d'un état dans un interféromètre	8
2.1.5	Probabilité de sortie : apparition du permanent	10
2.2	Description de l'interféromètre	14
2.3	Boson random sampling	17
2.3.1	Échantillonnage à l'aide de l'interféromètre	17
2.3.2	Analogie avec la planche de Galton	18
2.3.3	Intérêt du Boson random sampling	19
2.3.4	Analogie en Théorie des graphes	20
3	Complexité algorithmique	22
3.1	Classes de complexité des problèmes de décisions	22
3.1.1	P et NP	22
3.1.2	PP, BPP, BQP	23
3.1.3	Notion de difficulté, complet et exposant de complexité	24
3.2	Classe de complexité des problèmes de fonction et $\#P$	25
3.2.1	Problème de fonction	25
3.2.2	Classe $\#P$	26
3.3	Hiérarchie polynomiale et théorème associés	26
3.3.1	Hiérarchie polynomiale	26
3.3.2	Théorème de Toda	28
4	Théorème d'Aaronson & Arkhipov	29
4.1	Encoder une matrice unitaire dans un circuit linéaire optique	29
4.2	Encoder une matrice quelconque en matrice unitaire	29
4.3	Problème de l'exactitude du calcul du permanent	30
4.4	Théorème d'Aaronson	34
4.4.1	Complexité du calcul du permanent	34
4.4.2	Échantillonnage exact du Boson random sampling	38

1 Introduction

1.1 Prélude et contexte de travail

Mon travail de recherche s'inscrit dans le stage d'immersion, le projet d'intégration G3 et le mémoire de fin d'étude de mon Master de Mathématiques. Je réalise ce stage d'immersion à l'Inria dans l'équipe Paradyse. Cette équipe commune entre Inria, le CNRS et le Laboratoire Paul Painlevé, regroupe des mathématiciens de communautés différentes au service d'un même objectif : mieux comprendre la dynamique des systèmes de particules.

Le Laboratoire Paul Painlevé est une unité mixte de recherche en mathématiques (UMR 8524) sous la tutelle principale de l'Université de Lille (rattaché à la Faculté des Sciences et Technologies), le CNRS (rattaché à l'Institut National des Sciences Mathématiques et de leurs Interactions), et sous tutelle secondaire d'Inria. Les recherches menées au laboratoire couvrent la totalité du spectre de la recherche mathématique, tant fondamentale qu'appliquée.

L'Inria est l'institut national de recherche en sciences et technologies du numérique. L'Inria a pour mission le développement de la recherche et de la valorisation en sciences et techniques de l'information et de la communication, au niveau national comme au plan international.

Mon directeur de mémoire est Stephan de Bievre, mon professeur sur les Outils mathématiques pour la physique quantique.

Nous nous réunissons avec Stephan de Bievre et Christofer Langrenez, son chercheur en thèse, approximativement 1 à 2 fois par mois à l'Inria pour discuter de mes avancées et qu'ils puissent m'aider sur le sujet. De plus, j'ai eu l'occasion d'assister à des conférences d'experts sur l'informatique quantique et l'optique quantique. Enfin, j'assiste chaque Jeudi au cours sur les Outils mathématiques pour la physique quantique et aux cours de probabilités et de machine learning à Centrale qui me permettent de lier ce que j'apprends à mon sujet de mémoire.

1.2 Problème étudié

Les technologies quantiques espèrent mettre à profit les différences entre la mécanique classique et quantique afin d'exécuter de façon plus rapide, plus efficace ou plus sûre un certain nombre de tâches. C'est ce que l'on appelle l'avantage quantique. Le débat reste ouvert de savoir dans quelle mesure un tel avantage peut être atteint, dans quelles circonstances et pour quel type de tâche.

L'informatique quantique stimule depuis longtemps l'imagination des scientifiques et du public, qui

y voit la prochaine grande frontière, bien au-delà des supercalculateurs. S'appuyant sur des qubits plutôt que sur des bits classiques, utilisant les propriétés de la physique quantique pour stocker des données et effectuer des calculs, ces machines sont désormais construites à petite échelle.

La notion d'avantage quantique est un peu floue : par exemple, est-ce que le problème à résoudre doit être utile ou pas ? Ce que signifie résoudre un problème est clair : l'ordinateur quantique renvoie la bonne réponse en un temps raisonnable (en quelques heures ou quelques jours), mais il n'est pas clair de prouver qu'aucun algorithme ne peut résoudre ce même problème sur un ordinateur classique (peut-être qu'un bon algorithme n'a pas encore été trouvé).

Dès 2019, Google avait assuré avoir réalisé une telle prouesse avec un ordinateur quantique de 53 qubits, baptisé Sycamore, qui aurait réussi à effectuer une opération extrêmement complexe en moins de 4 min, quand les plus avancés des ordinateurs traditionnels auraient besoin de 10 000 ans pour y parvenir. Cet avancée avait été rapidement contestée par IBM, qui affirmait de son côté être venu à bout de l'opération sur une machine classique en deux jours et demi.

En 2021, IBM assénait un nouveau coup à la firme de Mountain View en présentant Eagle, premier système à passer le cap symbolique des 100 qubits. Estimant alors être proche de la suprématie quantique, IBM a dévoilé en novembre 2022 son processeur Osprey, muni cette fois de 433 qubits. À titre de comparaison, l'ordinateur quantique le plus avancé de Google recense pour l'instant 72 qubits. Pour les deux géants, qui affirment avoir atteint cette avantage quantique, on ne peut pas assurer scientifiquement que c'est vrai.

En 2011, Aaronson et Arkhipov ont introduit le Boson random sampling [1] et ont exploré l'utilisation possible de la diffusion des bosons pour évaluer des valeurs de permanent de matrice.

Le Boson random sampling est basé sur le principe de superposition quantique, qui permet aux photons de se trouver à plusieurs endroits à la fois. Cela signifie qu'un photon peut suivre plusieurs chemins différents en même temps, ce qui peut être utilisé pour effectuer des calculs de manière plus efficace que de manière classique. De plus, de nouveaux algorithmes ont été mis en oeuvre à partir du Boson random sampling pour réaliser des tâches irréalisable avec un ordinateur classique (des générateurs sans biais de nombre aléatoires, un algorithme de calculs de permanent de matrices définie semi positive [2] ...).

Il a suscité un grand intérêt dans le domaine de l'informatique quantique, car il offre un moyen de démontrer l'avantage de l'informatique quantique sur l'informatique classique pour certains types de calculs. Cependant, l'algorithme est également très difficile à mettre en oeuvre de manière pratique, car il nécessite une grande quantité de photons et un grand nombre de fonctions d'onde pour être exécuté correctement.

On est fortement convaincu que le Boson random sampling est une tâche informatique difficile à implémenter avec des ordinateurs classiques. Une configuration d'optique linéaire pour en faire un ordinateur quantique utiliserait beaucoup moins de ressources physiques qu'un ordinateur classique. Cet avantage en fait un candidat idéal pour démontrer la puissance du calcul quantique à court terme.

Dans ce mémoire, on expliquera comment le Boson random sampling peut être la solution d'un possible avantage quantique. On expliquera le modèle, les classes de complexité du calcul du permanent et expliquerons pourquoi il serait impossible de l'implémenter sur un ordinateur classique.

2 Boson Random Sampling

2.1 Formalisme mathématiques de l'optique quantique

Dans ce mémoire, nous nous concentrerons sur les bosons et plus particulièrement sur les photons. Ce sont les particules qui composent la lumière et avec lesquelles on peut travailler.

On se place dans le cadre où nous souhaitons décrire n photons. Pour les décrire, on peut travailler dans un espace de Hilbert \mathcal{H}_n des états à n photons. C'est un espace infini des états du système. Mais il est difficile de travailler directement sur cet espace : c'est pour cela qu'on introduit l'espace de Fock.

2.1.1 Décrire un état de la physique

Espace de Fock

L'espace de Fock est une construction algébrique utilisée en mécanique quantique pour construire l'espace des états quantiques. De manière informelle, un espace de Fock est la somme d'un ensemble d'espaces de Hilbert représentant zéro état de particule, un état de particule, deux états de particule, et ainsi de suite.

En mécanique quantique, un état de Fock ou état numérique est un état quantique qui est un élément d'un espace de Fock avec un nombre bien défini de particules (ou quanta).

On spécifie un état multiparticulaire de n particules identiques non interactives en écrivant l'état comme une somme de produits tensoriels de n états à une particule. Dans le cas des bosons, possédant un spin entier (et non régis par le principe d'exclusion) correspondent à des produits tensoriels symétriques.

Si nous souhaitons décrire m états pour n particules, l'espace de Fock se définit comme l'espace de Hilbert obtenu par la somme directe des produits tensoriels des espaces de Hilbert pour une particule.

Un état quantique dans l'espace de Fock peut s'écrire comme ceci : les $s_i \in \mathbb{N}$ représente le nombre de particule dans l'état i , et on écrit cet état : $|s_1 \dots s_m\rangle$, avec m le nombre d'état qu'il existe pour décrire 1 particule. Pour mieux comprendre, dans cet état de Fock, on a s_1 particules dans le premier état, s_2 particules dans le second état ... s_m particules dans le m^{me} état. Dans un système à n photons, $\sum_{i=1}^m s_i = n$.

Une base de l'espace de Fock appelée "base du nombre d'occupation". Un état quantique dans l'espace

de Fock est appelé état de Fock s'il est un élément de la base du nombre d'occupation.

Un état de Fock satisfait un critère important : pour chaque i , l'état est un état propre de l'opérateur nombre de particules dans l'état i N_i correspondant au i^{me} élément de l'état de Fock .

Ainsi, l'opérateur N_i agit comme ceci sur l'état de Fock :

$$N_i |s_1 \dots s_n\rangle = s_i |s_1 \dots s_n\rangle$$

Pour connaître le nombre de particule d'un état de Fock, on définit alors l'opérateur nombre de particule N qui est tout simplement la somme de tous les opérateurs N_i :

$$N |s_1 \dots s_n\rangle = n |s_1 \dots s_n\rangle$$

Par conséquent, pour travailler avec n photons dans m états, le tuple $S = (s_1, \dots, s_m)$ représentant les différents nombres d'occupations des photons dans les différents m états sont dans $\Phi_{m,n}$ qu'on définit :

$$\Phi_{m,n} = \{(s_1, s_2, \dots, s_m) : \sum_{j=1}^m s_j = n\}$$

Ainsi, $\{|S\rangle, S \in \Phi_{m,n}\}$ est l'ensemble des vecteurs représentant l'occupation de n photons dans m états quantiques .Cet ensemble forme une base de l'espace de Fock auquel on s'intéresse.

Superposition quantique

La particularité de la physique quantique par rapport à la physique classique est le principe de superposition : un même état quantique peut posséder plusieurs valeurs pour une certaine quantité physique observable.

Ce principe résulte du fait que l'état d'un système quantique est représenté par un vecteur dans un espace de Hilbert.

Ce vecteur admet une décomposition en une combinaison linéaire de vecteurs selon une base donnée. Or, il se trouve qu'en mécanique quantique, une observable donnée correspond à une base donnée de l'espace de Hilbert.

En conséquence, si l'on s'intéresse, par exemple, à l'occupation de n particule dans m états, l'état d'occupation doit être représenté comme une somme de vecteurs, chaque vecteur représentant les occupations des photons dans les m états de l'espace de Hilbert sous-jacent pour 1 particule.

Comme $\{|S\rangle, S = (s_1, s_2, \dots, s_m) \in \Phi_{m,n}\}$ forme une base de l'espace de Fock, on peut décrire tout état de l'espace comme ceci :

$$|\psi\rangle = \sum_{S \in \Phi_{m,n}} \alpha_S |S\rangle$$

avec $\alpha_S \in \mathbb{C}$ et $\sum_{S \in \Phi_{m,n}} |\alpha_S|^2 = 1$

Règle de Born

La règle de Born est une interprétation probabiliste des coefficients linéaires du principe de superposition.

$$|\psi\rangle = \sum_{S \in \Phi_{m,n}} \alpha_S |S\rangle$$

avec $\sum_{S \in \Phi_{m,n}} |\alpha_S|^2 = 1$.

Si on a un état $|\psi\rangle$, quand on va mesurer le nombre de photons qui occupe chaque état, on va mesurer un état qui est dans la combinaison linéaire du vecteur. La probabilité pour que le résultat de la mesure du nombre de photons dans chaque état soit $R = (r_1, r_2, \dots, r_m)$ est $|\alpha_R|^2$.

On peut alors définir cette probabilité comme ceci :

$$p_\psi(R) = |\langle R | \psi \rangle| = |\alpha_R|^2$$

2.1.2 Opérateur de création et d'annihilation pour les bosons

Les opérateurs de création et d'annihilation sont des opérateurs mathématiques qui ont de nombreuses applications en mécanique quantique, notamment dans l'étude des oscillateurs harmoniques quantiques et des systèmes à plusieurs particules.

Un opérateur d'annihilation (généralement noté a) diminue d'une unité le nombre de particules dans un état donné. Un opérateur de création (généralement noté a^\dagger) augmente de un le nombre de particules dans un état donné, et il est l'adjoint de l'opérateur d'annihilation.

Ainsi , $\forall |s_1 \dots s_n\rangle$, état de l'espace de Fock,

$$a_i |s_1 \dots s_n\rangle = \sqrt{s_i} |s_1 \dots s_i - 1 \dots s_n\rangle \quad (2.1)$$

$$a_i^\dagger |s_1 \dots s_n\rangle = \sqrt{s_i + 1} |s_1 \dots s_i + 1 \dots s_n\rangle \quad (2.2)$$

Avec ces définitions, on a les relations de commutation des opérateurs de création et d'annihilation suivantes :

$$[a_i, a_j^\dagger] = a_i a_j^\dagger - a_j^\dagger a_i = \delta_{ij}$$

$$[a_i^\dagger, a_j^\dagger] = [a_i, a_j] = 0$$

On peut définir l'opérateur nombre de particule dans l'état i à partir de ces opérateurs :

$$N_i = a_i^\dagger a_i$$

En effet, lorsque qu'on diminue de 1 puis augmente de 1 le nombre de photons dans l'état i , on revient bien à notre état de départ et la valeur propre de l'opérateur est bien le nombre s_i de particule dans l'état i .

2.1.3 Évolution dans le temps d'un état ou d'un opérateur

Les états que nous décrivons évoluent dans le temps. L'état à instant t $|\psi_t\rangle$ évolue selon l'équation de Schrödinger :

$$\begin{cases} i\hbar \frac{d}{dt} |\psi_t\rangle = H |\psi_t\rangle \\ \text{avec } |\psi_0\rangle = |\varphi\rangle \end{cases}$$

avec H l'opérateur hamiltonien du système. L'évolution dans le temps d'un système quantique est caractérisée (au niveau infinitésimal) par l'opérateur hamiltonien.

Ainsi, la solution de cette équation est :

$$|\psi_t\rangle = \exp\left(-\frac{iHt}{\hbar}\right) |\varphi\rangle$$

L'opérateur $U_t = \exp\left(-\frac{iHt}{\hbar}\right)$ est un opérateur unitaire qu'on appelle opérateur d'évolution.

C'est ce que l'on appelle la représentation de Schrödinger : on observe comment un état évolue dans le temps.

On peut définir la valeur moyenne d'un opérateur A quelconque pour un état $|\psi_t\rangle = U_t |\varphi\rangle$:

$$\begin{aligned} \langle A \rangle_{\psi_t} &= \langle \psi_t | A | \psi_t \rangle \\ &= \langle \varphi | U_{-t} A U_t | \varphi \rangle \end{aligned}$$

Dans cette interprétation, on remarque qu'on peut suivre l'évolution de la valeur moyenne de A avec les états : c'est l'interprétation de Schrödinger. On peut suivre aussi l'évolution de l'opérateur $U_{-t} A U_t$: c'est ce que l'on appelle la représentation d'Heisenberg.

L'évolution de l'opérateur est régie par l'équation de Heisenberg :

$$\frac{dA}{dt} = \frac{1}{i\hbar} [H, A]$$

Dans les 2 représentations, nous voyons bien que c'est l'Hamiltonien qui dicte l'évolution des objets de la mécanique quantique.

2.1.4 Modèle d'évolution d'un état dans un interféromètre

Modèle de l'interféromètre

Le modèle que nous allons étudier est celui de l'interféromètre en tant que circuit linéaire optique.

On considère n photons et m états. Les états d'entrées et de sorties sont dans l'espace :

$$\Phi_{m,n} = \{S = (s_1, s_2, \dots, s_m) : \sum_{j=1}^m s_j = n\}$$

En considérant a_j l'opérateur d'annihilation (*resp* a_j^\dagger l'opérateur de création), l'interféromètre réalise une transformation linéaire de l'opérateur de création [3] :

$$b_j^\dagger := \sum_{i=1}^m U_{ji} a_i^\dagger$$

avec U une matrice unitaire qui représente l'interféromètre. Nous reviendrons sur les éléments d'optique quantique dans une prochaine partie pour seulement s'intéresser ici aux mathématiques que nous avons besoin pour décrire le modèle.

On peut voir cette évolution des opérateurs d'échelle comme une évolution temporelle avec un temps infini. On se retrouve ainsi dans la représentation de Heisenberg. Les opérateurs b_j^\dagger sont alors les opérateurs de création après avoir évolué. Les opérateurs b_j^\dagger ont les mêmes propriétés de commutations que les opérateurs de création a_j^\dagger .

On a aussi le lemme suivant :

Lemme 1. *Il y a conservation du nombre de photons.*

Ainsi, le circuit linéaire optique réalise une transformation linéaire de l'opérateur de création et est stable pour tous les n sur les $\Phi_{m,n}$ à m fixé. Pour un état $|\psi_{in}\rangle$ d'entrée, on obtient un état de sortie $|\psi_{out}\rangle$ avec un opérateur unitaire $\varphi(U)$ relié à U qui agit sur les états :

$$|\psi_{out}\rangle = \varphi(U)|\psi_{in}\rangle$$

Ce $\varphi(U)$ est l'opérateur d'évolution de l'état du système dans la représentation de Heisenberg : on regarde à un temps ultérieur son évolution. De plus, on voit qu'ici, pour décrire l'évolution, on ne décrit pas explicitement l'Hamiltonien. Plus généralement, en optique quantique, on ne spécifie pas l'Hamiltonien. Il est en fait caché dans la matrice U .

Ainsi, on peut réécrire les opérateurs b_j^\dagger

$$b_j^\dagger = \varphi(U) a_j^\dagger \varphi(U)^\dagger$$

Pour transformer $|\psi_{in}\rangle$, on peut choisir une transformation unitaire $m \times m$ quelconque $U = (u_{ij})$. Cette matrice U induit une transformation unitaire plus grande $\varphi(U)$ sur l'espace de Hilbert \mathcal{H}_n des états à n photons. Il existe plusieurs façons de définir $\varphi(U)$, mais la plus simple est la formule suivante qui s'avère être la probabilité de sortie du circuit linéaire optique (à un module au carré près) que nous démontrerons par la suite :

$$\langle S | \varphi(U) | T \rangle = \frac{\text{Perm}(U_{S,T})}{\prod_{j=1}^m (s_j!)^{\frac{1}{2}} \prod_{i=1}^m (t_i!)^{\frac{1}{2}}}$$

pour tous les états $S = (s_1, \dots, s_m)$ et $T = (t_1, \dots, t_m)$ dans $\Phi_{m,n}$. $U_{S,T}$ est la matrice $n \times n$ obtenue à partir de U en prenant s_i copies de la i^e ligne de U et t_j copies de la j^e colonne, pour tout $i, j \in \llbracket 1, m \rrbracket$.

avec

$$\text{Perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i, \sigma_i}.$$

φ possède 2 propriétés :

- Pour U unitaire, $\varphi(U)$ est une transformation unitaire
- φ est un homomorphisme des matrices unitaires *i.e* $\forall U, W$ matrices unitaires, $\varphi(UW) = \varphi(U)\varphi(W)$

Preuve. Pour montrer que φ est un homomorphisme, on aimerai montrer que :

$$\langle S | \varphi(WU) | T \rangle = \langle S | \varphi(W) \varphi(U) | T \rangle = (*)$$

Par la relation de fermeture :

$$\begin{aligned} (*) &= \sum_{R \in \Phi_{m,n}} \langle S | \varphi(W) | R \rangle \langle R | \varphi(U) | T \rangle = \sum_{R \in \Phi_{m,n}} \frac{\text{Perm}(W_{S,R}) \text{Perm}(U_{R,T})}{\prod_{j=1}^m (s_j!)^{\frac{1}{2}} \prod_{i=1}^m (r_i!) \prod_{i=1}^m (t_i!)^{\frac{1}{2}}} \\ &= \prod_{j=1}^m (s_j!)^{-\frac{1}{2}} \prod_{i=1}^m (t_i!)^{-\frac{1}{2}} \sum_{R \in \Phi_{m,n}} \prod_{i=1}^m (r_i!)^{-1} \left(\sum_{\xi \in S_n} \prod_{i=1}^n (W_{S,R})_{i, \xi(i)} \right) \left(\sum_{\tau \in S_n} \prod_{j=1}^n (U_{R,T})_{j, \tau(j)} \right) \\ &= \prod_{j=1}^m (s_j!)^{-\frac{1}{2}} \prod_{i=1}^m (t_i!)^{-\frac{1}{2}} \sum_{R \in \Phi_{m,n}} \prod_{i=1}^m (r_i!)^{-1} \left(\sum_{\xi, \tau \in S_n} \prod_{i=1}^n (W_{S,R})_{i, \xi(i)} (U_{R,T})_{i, \tau(i)} \right) \\ &= \prod_{j=1}^m (s_j!)^{-\frac{1}{2}} \prod_{i=1}^m (t_i!)^{-\frac{1}{2}} \sum_{\sigma \in S_n} \prod_{i=1}^n ((WU)_{S,T})_{i, \sigma(i)} \end{aligned}$$

Pour expliquer ce dernier passage, nous avons décomposé la somme en considérant que chaque permutation $\sigma \in S_n$ est un produit de deux permutations : l'une, ξ , qui fait correspondre n particules dans la configuration initiale $|S\rangle$ à n particules dans la configuration intermédiaire $|R\rangle$ lorsque W est appliqué, et une autre, τ , qui associe n particules dans la configuration intermédiaire $|R\rangle$ à n particules

dans la configuration finale $|T\rangle$ lorsque U est appliquée.

On obtient ce résultat, tant que l'on se souvient de faire la somme de toutes les configurations intermédiaires possibles $R \in \Phi_{m,n}$, et aussi de diviser chaque somme par $r_1! \dots r_n!$, qui est la taille du groupe d'automorphisme de R (c'est-à-dire le nombre de façons de permuter les n particules dans $|R\rangle$ qui laissent $|R\rangle$ inchangé).

On obtient alors :

$$\begin{aligned} \langle S | \varphi(W) \varphi(U) | T \rangle &= \prod_{j=1}^m (s_j!)^{-\frac{1}{2}} \prod_{i=1}^m (t_i!)^{-\frac{1}{2}} \text{Perm}((WU)_{S,T}) \\ &= \langle S | \varphi(WU) | T \rangle \end{aligned}$$

□

2.1.5 Probabilité de sortie : apparition du permanent

Positionnement du problème

Pour réécrire un état du système avec s_i photons à l'entrée i , sachant qu'il y a n photons et m états, on décrit ce vecteur d'entrée $|\psi_{in}\rangle = |s_1 s_2 \dots s_m\rangle$.

Dans l'état de départ, il y a n photons. Ainsi, l'espace de départ est :

$$\Phi_{m,n} = \{(s_1, s_2, \dots, s_m) : \sum_{j=1}^m s_j = n\}$$

Cette espace est stable par une transformation unitaire : il y a conservation du nombre de photons par transformation unitaire. En effet, dans le modèle, on considère qu'il y a ni création ni destruction de photons après le passage de l'état dans le circuit optique. [4]

On peut considérer le circuit linéaire optique comme un opérateur unitaire $V = \varphi(U)$ qui s'applique à un état d'entrée :

$$|\psi_{out}\rangle = V|\psi_{in}\rangle$$

Ce qui nous intéresse est la probabilité de sortie et d'après la règle de Born :

$$P_U(\text{in} = |\psi_{in}\rangle, \text{out} = |t_1 t_2 \dots t_m\rangle) = |\langle t_1 t_2 \dots t_m | \psi_{out} \rangle|^2$$

Or

$$\langle t_1 t_2 \dots t_m | \psi_{out} \rangle = \langle t_1 t_2 \dots t_m | V | \psi_{in} \rangle = \langle 0 | \prod_{i=1}^m \frac{a_i^{t_i}}{\sqrt{t_i!}} V | \psi_{in} \rangle$$

Si on a zéro photon à l'entrée du circuit, on a zéro photon à la sortie et donc $V|0\rangle = |0\rangle$ et $V^\dagger|0\rangle = |0\rangle$.

De plus, comme $V^\dagger V = I$, on peut rajouter l'opérateur V et V^\dagger d'un coté et de l'autre des coefficients multiplicatifs. Ils se simplifient en opérateur identité dans les multiplications et on a :

$$V^\dagger \left(\prod_{i=1}^m \frac{a_i^{t_i}}{\sqrt{t_i!}} \right) V = \prod_{i=1}^m \frac{(V^\dagger a_i V)^{t_i}}{\sqrt{t_i!}}$$

Ainsi , en définissant $b_j := V^\dagger a_j V$ et $|\phi_T\rangle = \prod_{i=1}^m \frac{(b_i^\dagger)^{t_i}}{\sqrt{t_i!}} |0\rangle$:

$$\begin{aligned} \langle T | \psi_{out} \rangle &= \langle 0 | V V^\dagger \left(\prod_{i=1}^m \frac{a_i^{t_i}}{\sqrt{t_i!}} \right) V | \psi_{in} \rangle \\ &= \langle 0 | \prod_{i=1}^m \frac{(V^\dagger a_i V)^{t_i}}{\sqrt{t_i!}} | \psi_{in} \rangle \\ &= \langle 0 | \prod_{i=1}^m \frac{b_i^{t_i}}{\sqrt{t_i!}} | \psi_{in} \rangle \\ &= \langle \phi_T | \psi_{in} \rangle \end{aligned}$$

Par conséquent, d'après la définition du modèle du Boson random sampling , cela nous revient à calculer [5] :

$$|\phi_T\rangle = \prod_{j=1}^m (b_j^\dagger)^{t_j} |0\rangle = \prod_{j=1}^m \frac{1}{\sqrt{t_j!}} \left(\sum_{i=1}^m U_{ji} a_i^\dagger \right)^{t_j} |0\rangle$$

Apparition du permanent

La probabilité de mesurer $|T\rangle = |t_1 t_2 \dots t_m\rangle$ avec une entrée $|S\rangle = |s_1 s_2 \dots s_m\rangle$ dans la transformation unitaire U a été démontré pour la première fois par Sheel [5]

Lemme 2.

$$P_U(S, T) = |\langle T | \psi_{out} \rangle|^2 = |\langle \phi_T | S \rangle|^2 = \frac{|\text{Perm}(U_{S,T})|^2}{\prod_{j=1}^m (s_j!) \prod_{i=1}^m (t_i!)}$$

Pourquoi le permanent ? Exemple avec 2 photons

Soit $U = \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix}$ matrice représentant le circuit.

On suppose que U est unitaire. Comme nous avons pu le définir plus haut, le Boson random sampling réalise une transformation linéaire des opérateurs d'échelle. Ainsi, comme la matrice U est de taille (2×2) :

$$b_j^\dagger = U_{j1}a_1^\dagger + U_{j2}a_2^\dagger$$

Comme décrit plus haut, nous avons besoin de calculer le vecteur $|\phi_T\rangle$ en fonction de la sortie $|t_1 t_2\rangle$. Prenons pour cet exemple $|t_1 t_2\rangle = |11\rangle$ et calculons son vecteur " $|\phi_T\rangle$ " associé :

$$\begin{aligned} |\phi_T\rangle &= b_1^\dagger b_2^\dagger |00\rangle = (U_{11}a_1^\dagger + U_{12}a_2^\dagger)(U_{21}a_1^\dagger + U_{22}a_2^\dagger)|00\rangle \\ &= (U_{11}U_{21}a_1^{\dagger 2} + U_{11}U_{22}a_1^\dagger a_2^\dagger + U_{12}U_{21}a_2^\dagger a_1^\dagger + U_{12}U_{22}a_2^{\dagger 2})|00\rangle \end{aligned}$$

On sait que $[a_i^\dagger; a_j^\dagger] = 0$ et donc que $a_2^\dagger a_1^\dagger = a_1^\dagger a_2^\dagger$. Donc, on peut regrouper les termes et :

$$\begin{aligned} |\phi_T\rangle &= (U_{11}U_{21}a_1^{\dagger 2} + (U_{11}U_{22} + U_{12}U_{21})a_1^\dagger a_2^\dagger + U_{12}U_{22}a_2^{\dagger 2})|00\rangle \\ &= \sqrt{2}U_{11}U_{21}|20\rangle + (U_{11}U_{22} + U_{12}U_{21})|11\rangle + \sqrt{2}U_{12}U_{22}|02\rangle \end{aligned}$$

D'après la règle de Born, on obtient les résultats de probabilités de sorties du circuit :

$$\begin{aligned} |\langle\phi_T|20\rangle|^2 &= 2|U_{11}|^2|U_{21}|^2 = \frac{1}{2} \left| \text{Perm} \begin{pmatrix} U_{11} & U_{11} \\ U_{21} & U_{21} \end{pmatrix} \right|^2 \\ |\langle\phi_T|02\rangle|^2 &= 2|U_{12}|^2|U_{22}|^2 = \frac{1}{2} \left| \text{Perm} \begin{pmatrix} U_{12} & U_{22} \\ U_{12} & U_{22} \end{pmatrix} \right|^2 \\ |\langle\phi_T|11\rangle|^2 &= \left| \text{Perm} \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \right|^2 \end{aligned}$$

On remarque alors que le permanent d'une certaine matrice $U_{S,T}$, dépendant de l'entrée et de la sortie du système, apparaît. On peut faire les calculs pour toutes les sorties mais on voit bien que la transformation linéaire des opérateurs de créations transforme les vecteurs d'entrées en combinaison linéaire d'états de Fock ayant en coefficient scalaire un produit de termes de la matrice U .

Cas général

Preuve. Partons désormais du fait que nous avons à calculer :

$$|\phi_T\rangle = \prod_{j=1}^m \frac{1}{\sqrt{t_j!}} \left(\sum_{i=1}^m U_{ji} a_i^\dagger \right)^{t_j} |0\rangle$$

Maintenant, nous utilisons le théorème d'expansion multinomiale : $\forall k, m \in \mathbb{N}, \forall (x_1, \dots, x_m) \in \mathbb{K}$,

$$\left(\sum_{i=1}^m x_i \right)^k = \sum_{\substack{\{k_n\} \\ \sum_n k_n = k}} \binom{k}{k_1, \dots, k_m} \prod_{i=1}^m x_i^{k_i}$$

avec

$$\binom{k}{k_1, \dots, k_m} = \frac{k!}{\prod_i (k_i!)}$$

.

Par conséquent, en appliquant cette formule sur $x_i = U_{ji}a_i^\dagger$ et $k = t_j$:

$$\left(\sum_{i=1}^m U_{ji}a_i^\dagger\right)^{t_j} = \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \binom{t_j}{t_{1j}, \dots, t_{mj}} \prod_{i=1}^m (U_{ji}a_i^\dagger)^{t_{ij}}$$

Et nous obtenons :

$$\begin{aligned} |\phi_T\rangle &= \prod_{j=1}^m \frac{1}{\sqrt{t_j!}} \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \binom{t_j}{t_{1j}, \dots, t_{mj}} \prod_{i=1}^m (U_{ji}a_i^\dagger)^{t_{ij}} |0\rangle \\ &= \prod_{j=1}^m \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\sqrt{t_j!}}{\prod_i t_{ij}!} \prod_{i=1}^m (U_{ji}a_i^\dagger)^{t_{ij}} |0\rangle \end{aligned}$$

Maintenant, on intervertit la somme et le produit avec la formule suivante :

$$\prod_{i=1}^m \left(\sum_{j=1}^n c_{ij}\right) = \sum_{f \in \mathcal{F}_{m,n}} \prod_{i=1}^m c_{i,f(i)}$$

avec $\mathcal{F}_{m,n} = \{f, f : \llbracket 1, m \rrbracket \rightarrow \llbracket 1, n \rrbracket\}$

Et on obtient :

$$|\phi_T\rangle = \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\prod_{j=1}^m (t_j!)^{\frac{1}{2}}}{\prod_{i,j=1}^m t_{ij}!} \prod_{i=1}^m \left(\prod_{j=1}^m (U_{ji}a_i^\dagger)^{t_{ij}}\right) |0\rangle$$

Maintenant qu'on a cela, on sait que :

$$\begin{cases} \sum_i t_{ij} = t_j \text{ représente le nombre de photons à la sortie} \\ \sum_j t_{ij} = s_i \text{ représente le nombre de photons à l'entrée} \end{cases}$$

Et on peut réécrire la ligne du dessus comme ceci en appliquant les différents opérateur d'échelle :

$$|\phi_T\rangle = \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\prod_{i=1}^m (t_i!)^{\frac{1}{2}}}{\prod_{i,j=1}^m t_{ij}!} \left(\prod_{i,j} (U_{ji})^{t_{ij}} \right) \prod_{i=1}^m \sqrt{(\sum_j t_{ij})!} \sum_j t_{1j} \sum_j t_{2j} \cdots \sum_j t_{mj} \rangle \quad (2.3)$$

$$= \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\prod_{i=1}^m (s_i!)^{\frac{1}{2}} \prod_{i=1}^m (t_i!)^{\frac{1}{2}}}{\prod_{i,j=1}^m t_{ij}!} \left(\prod_{i,j} (U_{ji})^{t_{ij}} \right) |s_1 s_2 \dots s_m\rangle \quad (2.4)$$

Pour un choix de $(t_{ij})_{i,j \in \{1 \dots m\}}$, on a un facteur en produit de coefficient de la matrice U à la puissance t_{ij} . Si on somme sur toutes les possibilités des $(t_{ij})_{i,j \in \{1 \dots m\}}$, on obtient le permanent d'une matrice.

Pour une entrée $|s_1 s_2 \dots s_m\rangle$ et une sortie $|t_1 t_2 \dots t_m\rangle$, la matrice en question, qu'on appellera $U_{S,T}$ est la matrice obtenu en répétant s_j fois la i^{me} colonne et t_j fois la j^{me} ligne de la matrice U .

Pour être plus précis, et en utilisant les notations introduise par Scheel [5], $U_{S,T} = U[(s_1 s_2 \dots s_m), (t_1 t_2 \dots t_m)] = (U_{s_i, t_j})_{i,j \in \{1 \dots m\}}$.

Ainsi pour un $|\psi_{in}\rangle = |s_1 s_2 \dots s_m\rangle$

$$\langle t_1 t_2 \dots t_m | \psi_{out} \rangle = \langle \phi_T | \psi_{in} \rangle = \frac{\text{Perm}(U_{S,T})}{\prod_{j=1}^m (s_j!)^{\frac{1}{2}} \prod_{i=1}^m (t_i!)^{\frac{1}{2}}}$$

et la probabilité de mesurer $|t_1 t_2 \dots t_m\rangle$ avec une entrée $|s_1 s_2 \dots s_m\rangle$ dans la transformation unitaire U est :

$$P_U(S, T) = |\langle t_1 t_2 \dots t_m | \psi_{out} \rangle|^2 = \frac{|\text{Perm}(U_{S,T})|^2}{\prod_{j=1}^m (s_j!) \prod_{i=1}^m (t_i!)}$$

Ainsi, pour une entrée S fixée, on obtient une densité de probabilité ne dépendant que de U et se calcule à l'aide d'un calcul de permanent. Dans le modèle du Boson random sampling, on fixe l'entrée : on considérera n photons qui sont chacun dans un mode différent parmi les m états et ce seront les n premiers états. L'état de Fock d'entrée sera $|s_1 s_2 \dots s_m\rangle = |1_n\rangle = |11 \dots 100 \dots 0\rangle$

□

2.2 Description de l'interféromètre

Physiquement, $\varphi(U)$ est unitaire parce qu'il représente une transformation physique réelle qui peut être appliquée, et φ est un homomorphisme parce que la généralisation d'un photon à n photons doit commuter avec la composition des séparateurs de faisceaux.

Pour bien comprendre ce qu'est un interféromètre, imaginons que nous ayons un état avec une seule particule. Nous pouvons alors représenter notre état par

$$\sum_{i=1}^m \alpha_i |i\rangle$$

où $|i\rangle = |0, \dots, 1, \dots, 0\rangle$ est l'état de Fock qui a un photon qui occupe le i^{me} état. Pour changer cet état, il va passer à travers un interféromètre, qui sera ici un réseau d'éléments optiques les plus simples qui sont les déphaseurs (*phase-shifters*) et les séparateurs de faisceaux (*beamsplitters*).

La particule occupera un des états. Dans ce contexte, ces états sont les modes optiques d'une onde électromagnétique.

Pour rappel, un mode est un état d'une onde stationnaire qui est solution de l'équation de Schrödinger. Ainsi, la structure de ces modes est régit seulement par l'opérateur Hamiltonien qui décrit l'évolution dans le temps des états. Chaque mode ici est un oscillateur harmonique quantique et ses oscillations effectuées sont sinusoïdales, avec une amplitude et une fréquence qui ne dépendent que des caractéristiques intrinsèques du système, c'est à dire l'Hamiltonien, et des conditions initiales.

Pour construire un circuit linéaire, il nous faut :

- Une source de photon
- Des beamsplitters et des phase-shifters
- Un détecteur de photon

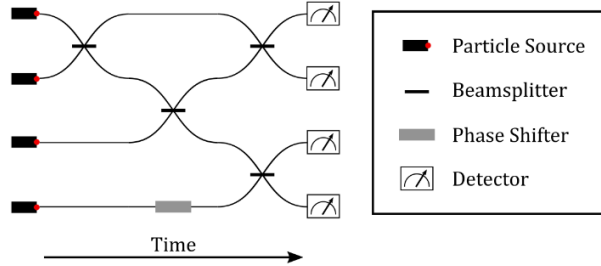


FIGURE 2.1 – Interféromètre

Beamsplitters

Les beamsplitters agissent sur deux modes - en pratique, souvent sur deux modes adjacents. Un séparateur de faisceaux peut être représenté par une matrice unitaire 2×2 . Les expérimentateurs se limitent généralement à des matrices réelles de la forme :

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

La matrice $m \times m$ correspondante (en appliquant R_θ sur les deux premiers modes) serait la somme directe :

$$B_\theta = U = \begin{pmatrix} R_\theta & \\ & I_{m-2} \end{pmatrix}$$

Phase-shifters

Les déphaseurs appliquent un déphasage complexe à un mode spécifique. En tant que matrices unitaires $m \times m$, il s'agit de matrices diagonales de la forme

$$\begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & e^{i\varphi} & \\ & & & \ddots \\ & & & & 1 \end{pmatrix}$$

où la i^{me} entrée, agissant sur le i^{me} mode, est $e^{i\varphi}$, et toutes les autres entrées diagonales sont à 1.

Construire un interféromètre optique

On souhaiterait réaliser une transformation linéaire des opérateurs de création et d'annihilation pour simuler physique la transformation du modèle que nous avons expliqué :

$$b_j^\dagger := \sum_{i=1}^m U_{ji} a_i^\dagger$$

On veut alors "encoder" une matrice U dans un circuit linéaire optique. Le théorème suivant nous donne l'existence d'un tel encodage [**Reck**] :

Théoreme 1. *Soit U une matrice unitaire de taille (m, m) . On peut réaliser un circuit linéaire optique représentant cette matrice U avec $O(m^2)$ beamsplitters et phase-shifters.*

La preuve de ce théorème se résume au fait que le pivot de Gauss sur une matrice $m \times m$ nécessite $O(m^2)$ opérations de rangée, et nous pouvons intégrer n'importe quelle opération de rangée dans des beamsplitters et des phase-shifters. On peut comparer cela au fait que 4^m portes sont nécessaires pour produire des unitaires $m \times m$ arbitraires dans un modèle de calcul par qubits.

Il y a donc un algorithme récursif qui factorise toute matrice unitaire en une séquence de transformations de séparateur de faisceau bidimensionnel. Ainsi, des expériences optiques avec tout type de rayonnement (photons, atomes, etc.) explorant des systèmes quantiques discrets de plus grande dimension deviennent réalisables.

2.3 Boson random sampling

2.3.1 Échantillonnage à l'aide de l'interféromètre

Construction physique Considérons un circuit optique linéaire multimode de m modes qui est injecté avec n photons uniques indiscernables ($m > n$). La mise en œuvre photonique de l'échantillonnage de bosons consiste à générer un échantillon à partir de la distribution de probabilité des mesures de photon unique en sortie du circuit. Plus précisément, cela nécessite des sources fiables de photons uniques, ainsi qu'un interféromètre linéaire. Ce dernier peut être fabriqué, par exemple, avec des séparateurs de faisceau à fibres fusionnées, par silice sur silicium ou écrit au laser des interféromètres intégrés, ou des puces optiques à interface électrique et optique.

Enfin, le schéma nécessite également des détecteurs de comptage de photons uniques à haut rendement, tels que ceux basés sur des nanofils supraconducteurs polarisés en courant, qui effectuent les mesures à la sortie du circuit. Par conséquent, sur la base de ces trois ingrédients, la configuration d'échantillonnage de bosons ne nécessite aucun bit auxiliaire, mesure adaptative ou opération d'intrication.

Cela en fait un modèle non universel de calcul quantique et réduit la quantité de ressources physiques nécessaires à sa réalisation pratique

Construction mathématique du Boson random sampling

Soit U une matrice unitaire. On peut construire un interféromètre qui encode U et qui réalise la transformation des opérateurs de création.

Pour une entrée $|S\rangle = |s_1 \dots s_m\rangle$, on aura en sortie l'état $\varphi(U)|S\rangle$ qui est une superposition d'état de Fock :

$$\varphi(U)|S\rangle = \sum_{T \in \Phi_{m,n}} \langle T | \varphi(U) | S \rangle |T\rangle =$$

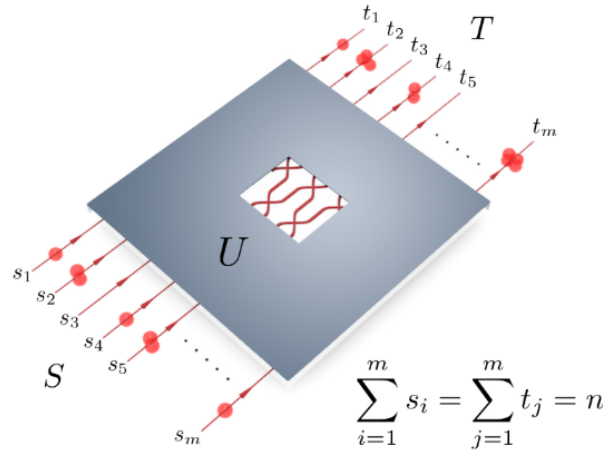


FIGURE 2.2 – Circuit linéaire optique

Ensuite nous mesurons à la sortie le nombre de photons dans chaque modes. On sait que cette sortie est une superposition et que la probabilité de mesurer la configuration d'occupation T à la sortie est :

$$P_U(S, T) = |\langle T | \varphi(U) | S \rangle|^2 = \frac{|\text{Perm}(U_{S,T})|^2}{\prod_{j=1}^m (s_j!) \prod_{i=1}^m (t_i!)}$$

Ainsi, en fixant l'entrée du système avec par exemple n photons dans les n premiers modes parmi les m modes et en effectuant l'expérience plusieurs fois, on simule une densité de probabilité dépendant du permanent des $U_{S,T}$.

Le Boson random sampling est tout simplement le fait d'arriver à simuler physiquement cette densité de probabilité en échantillonnant à l'aide du système que nous avons décrit.

2.3.2 Analogie avec la planche de Galton

Le Boson Random Sampling est similaire au *Galton Board* [1].

La table de Galton est un dispositif inventé par Sir Francis Galton qui illustre la convergence de la loi binomiale avec la loi normale.

Les clous sont enfoncés sur la planche de sorte qu'une boule lâchée sur la planche aille à droite ou à gauche pour chaque rangée de clous. En bas, les balles sont regroupées par le nombre de passes gauche et droite effectuées.

Ainsi, chaque case correspond à une issue possible de l'expérience binomiale (comme une expérience de Bernoulli répétée) et on voit que la répartition des boules dans les cases se rapproche d'une courbe gaussienne, d'autant plus que le nombre de lignes augmente ; autrement dit : la distribution binomiale converge vers la distribution normale. Ceci est donc une illustration du théorème de Moivre-Laplace.

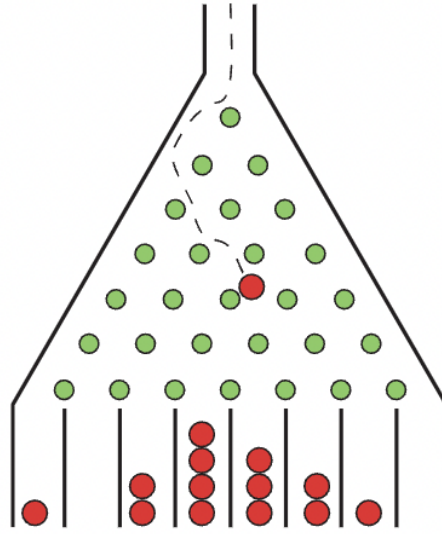


FIGURE 2.3 – Galton Board

On peut voir le Boson random sampling comme une planche de Galton avec :

- Les boules sont des photons
- Les clous sont des beamsplitters
- L'ensemble des clous est la matrice U unitaire
- La loi normale est la densité de probabilité avec les permanents

2.3.3 Intérêt du Boson random sampling

La principale raison de l'intérêt croissant pour le modèle d'échantillonnage de bosons est que, malgré son caractère non universel, on pense fortement qu'il permet d'effectuer une tâche de calcul impossible à réaliser par un ordinateur classique mais peut être par un ordinateur quantique réel. L'une des principales raisons en est que la distribution de probabilité, que le dispositif d'échantillonnage de bosons doit échantillonner, est liée comme on peut le voir au permanent des matrices complexes.

Le calcul du permanent est, dans le cas général, une tâche extrêmement difficile : il tombe dans la classe de complexité $\#P$ -hard. De plus, son approximation à l'erreur multiplicative près est également un problème $\#P$ -hard. Nous expliqueront ces termes de complexité dans une prochaine partie, mais nous pouvons dire à ce moment que calculer un permanent, c'est très difficile.

Ainsi, en échantillonnant plusieurs fois un circuit de Boson sampling, on peut s'attendre à approximer la densité de probabilité créée par l'opérateur U du circuit optique. Néanmoins, bien que la probabilité $P_U(S, T)$ d'un résultat de mesure spécifique à la sortie de l'interféromètre est liée au permanent d'une sous matrice de U , une matrice unitaire, une machine d'échantillonnage de bosons ne permet pas son estimation.

La raison principale est que la probabilité de détection correspondante est généralement exponentiellement faible. Ainsi, afin de collecter suffisamment de statistiques pour approximer sa valeur, il faut exécuter l'expérience quantique pendant une durée exponentiellement longue. Par conséquent, l'estimation obtenue à partir d'un échantillonneur de bosons n'est pas plus efficace que l'exécution de l'algorithme classique.

Ce qu'il faut bien comprendre, c'est que malgré le fait que le Boson random sampling n'est pas plus efficace pour calculer un permanent, il est cependant efficace pour générer une distribution irréalisable (sous réserve de $P \neq NP$) sur un ordinateur classique. Sans le détail complet pour l'instant, Aaronson et Arkhipov ont montré que si un algorithme classique peut échantillonner des états de Fock à partir de la même distribution de probabilité que peut "générer" le Boson sampling, c'est comme si $P = NP$.

2.3.4 Analogie en Théorie des graphes

Le calcul du permanent peut s'avérer important en théorie des graphes.

Soit G un graphe bipartie $m \times m$. On appelle la fonction pm la fonction qui assigne à un graphe bipartie son nombre de match parfait. Le graphe G définit deux parties $E = \{e_1, \dots, e_m\}$ et $F = \{f_1, \dots, f_m\}$ et A la matrice d'incidence du graphe. Il y a une arête entre $u_j \in U$ et $v_j \in V$ si et seulement si $a_{ij} = 1$. Sinon, il n'y a pas d'arête si et seulement si $a_{ij} = 0$. Alors :

$$\text{pm}(G) = \text{Perm}(A)$$

La matrice A donne l'information de comment sont connectés les sommets entre les deux parties E et F et son permanent est le nombre de match parfait.

Le Boson random sampling et ce problème de théorie de graphe sont liés en considérant l'ensemble E qui est l'ensemble des modes d'entrées du Boson sampling et F l'ensemble des états de sorties après l'application d'une matrice unitaire U . Ainsi, le permanent de la matrice $U_{S,T}$ est la somme des poids totaux (produit des poids des arêtes) pour tous les appariements parfaits du graphe biparti. Le produit des poids des arêtes d'un match représente l'importance du match en terme de probabilité. Par conséquence plus les match sont importants, plus cette permutation sera probable par les photons [feder].

Dans l'exemple avec $m = 2$ et $n = 2$, $b_1^\dagger b_2^\dagger |00\rangle = \sqrt{2}U_{11}U_{21}|20\rangle + (U_{11}U_{22} + U_{12}U_{21})|11\rangle + \sqrt{2}U_{12}U_{22}|02\rangle$

Le coefficient devant l'état $|20\rangle(\sqrt{2}U_{11}U_{21})$ indique que pour passer d'un photon dans l'état 1 et d'un photon dans l'état 2 (état $|11\rangle$), il faudra que le photon dans l'état 1 reste dans l'état 1 et que le photon dans l'état 2 passe dans l'état 1.

Le coefficient devant l'état $|02\rangle(\sqrt{2}U_{12}U_{22})$ indique que pour passer d'un photon dans l'état 1 et d'un photon dans l'état 2 (état $|11\rangle$), il faudra que le photon dans l'état 1 passe dans l'état 2 et que le photon dans l'état 2 reste dans l'état 2.

Le coefficient devant l'état $|11\rangle((U_{11}U_{22} + U_{12}U_{21}))$ indique que pour passer d'un photon dans l'état 1 et d'un photon dans l'état 2 (état $|11\rangle$) , il y a deux possibilité :

- Soit le photon dans l'état 1 reste dans l'état 1 et le photon dans l'état 2 reste dans l'état 2.
- Soit le photon dans l'état 1 passe dans l'état 2 et le photon dans l'état 2 passe dans l'état 1.

Le calcul explicite de l'amplitude des différents états nous donnent les différentes possibilités pour qu'un état d'entrée de Fock deviennent un état de sortie de Fock.

3 Complexité algorithmique

Dans ce mémoire, on s'intéressera à la complexité des algorithmes car c'est avec ces notions que nous pouvons décrire la difficulté d'un problème informatique, que ce soit en classique et en quantique.

On appelle complexité d'un algorithme son temps de calcul (autrement dit le temps nécessaire à son exécution). On s'intéresse à l'évolution de cette complexité en fonction de la taille des données ; autrement dit, étant donnée une procédure p , on s'intéresse à la variation de la durée du calcul de $p(x)$ en fonction de la taille de l'argument x .

Cette définition de la complexité semble reposer sur une définition précise du temps de calcul ; or tout le monde sait que celui-ci, exprimé par exemple en secondes, dépend de la machine utilisée (vitesse du processeur, temps d'accès à la mémoire, etc.), et décroît rapidement avec les progrès de la technologie. Pour se rapprocher d'une notion indépendante de la technologie, on adopte la convention suivante :

Le temps de calcul de $p(x)$ est le nombre d'instructions élémentaires exécutées pendant ce calcul.

La définition d'une instruction élémentaire dépend à son tour du modèle de calcul considéré, mais un consensus est facile à trouver en pratique ; en cas de doute, on se ramène à une estimation du nombre d'instructions effectuées par le processeur. Une autre solution, employée en particulier pour démontrer les théorèmes fondamentaux de la théorie de la complexité, est d'utiliser le modèle des machines de Turing, pour lequel la définition d'une instruction élémentaire (appelée aussi transition) est non ambiguë.

Ainsi il existe plusieurs classes de complexité : $P, NP, BPP, BQP, \#P$ etc. Ces classes de complexité s'intéressent aux problèmes de décision et aux problèmes de comptage. Pour la suite, on notera $\{0, 1\}^* = \{0, 1\}, \{0, 1\}^2, \dots\} = \bigcup_{n \in \mathbb{N}} \{0, 1\}^n$

3.1 Classes de complexité des problèmes de décisions

3.1.1 P et NP

Par commodité, on considère le plus souvent des problèmes de décision, où la tâche est de décider si une donnée $x \in \{0, 1\}^*$ est dans un langage dit $L \subset \{0, 1\}^*$, qui est un ensemble de chaînes de bits. Une machine qui calcule la fonction booléenne $f_L : \{0, 1\}^* \rightarrow \{0, 1\}$, qui satisfait $f_L(x) = 1 \iff x \in L$, décide L . Par exemple, un langage L pourrait être donné par l'ensemble de tous les graphes pour lesquels il existe un chemin qui visite chaque sommet une fois, en codage binaire, et une chaîne $x \in L$

est le codage binaire d'une instance particulière du graphe.

Définition 1 (P). *Un langage $L \subset \{0, 1\}^*$ est dans la classe P s'il existe un algorithme classique \mathcal{A} qui, étant donné $x \in \{0, 1\}^*$ en entrée, décide si $x \in L$ en temps d'exécution polynomial en $|x|$:*

$$x \in L \iff \mathcal{A}(x) = 1$$

Cette classe est une classe de complexité très importante. Un problème est dans la classe P s'il est décidé en temps polynomial par rapport à la taille de l'entrée. On dit que le problème est décidé en temps polynomial.

Les problèmes dans P sont considérés comme « faisables », faciles à résoudre (dans le sens où on peut le faire relativement rapidement). On peut le réécrire comme ceci :

$$P = \bigcup_{c \leq 1} \text{TIME}(n^c)$$

Définition 2 (NP). *Un langage $L \subset \{0, 1\}^*$ est dans la classe NP s'il existe un polynôme $p : \mathbb{N} \rightarrow \mathbb{N}$ et un algorithme classique en temps polynomial \mathcal{V} (appelé le vérificateur de L) tel que pour tout $x \in \{0, 1\}^*$,*

$$x \in L \iff \exists y \in \{0, 1\}^{p(|x|)} : \mathcal{V}(x, y) = 1$$

.

L'abréviation NP signifie « non déterministe polynomial ». Cela ne veut pas dire que ce n'est pas polynomiale. Un problème de décision est dans NP s'il est décidé par un algorithme non déterministe en temps polynomial par rapport à la taille de l'entrée. Intuitivement, cela revient à dire qu'on peut vérifier « rapidement » (complexité polynomiale) si une solution candidate est bien solution.

P et NP sont les classes les plus importantes de la théorie de la complexité qui soulève un des problèmes les plus difficiles du millénaire. L'Institut de mathématiques Clay a inclus ce problème dans sa liste des sept problèmes du prix du millénaire, et offre à ce titre un million de dollars à quiconque sera en mesure de démontrer $P = NP$ ou $P \neq NP$ ou de démontrer que ce n'est pas démontrable.

Très schématiquement, il s'agit de déterminer si le fait de pouvoir vérifier rapidement une solution à un problème implique de pouvoir la trouver rapidement ; ou encore, si ce que nous pouvons trouver rapidement lorsque nous avons de la chance peut être trouvé aussi vite par un calcul intelligent.

Néanmoins, les experts du domaine de la complexité penchent plus pour que $P \neq NP$ et nous prendrons cette information importante pour décrire le théorème d'Aaronson & et Arkhipov

3.1.2 PP, BPP, BQP

Pour comparer la théorie de la complexité des algorithmes classiques versus quantiques, il faut ajouter dans les définitions une part de probabilité et de hasard. Ainsi, on compare des algorithmes quantiques avec des algorithmes classiques randomisés ajoutant une part d'aléatoire.

Définition 3 (PP). On appelle *PP* la classe de tous les langages $L \subset \{0, 1\}^*$ pour lesquels il existe un algorithme classique randomisé \mathcal{A} qui décide en temps polynomial tel que pour tout $n \in \mathbb{N}$ et toutes les entrées $x \in \{0, 1\}^n$

$$x \in L \Rightarrow \Pr_r[\mathcal{A}(x, r) = 1] > 1/2$$

$$x \notin L \Rightarrow \Pr_r[\mathcal{A}(x, r) = 1] < 1/2$$

où la probabilité est prise sur l'aléa interne de l'algorithme.

C'est une classe de complexité probabiliste. Plus précisément c'est l'ensemble de problèmes de décision probabiliste solvable en temps polynomial avec une probabilité d'erreur inférieure à un demi.

Définition 4 (BPP (BQP)). On appelle *PP* la classe de tous les langages $L \subset \{0, 1\}^*$ pour lesquels il existe un algorithme classique randomisé \mathcal{A} qui décide en temps polynomial tel que pour tout $n \in \mathbb{N}$ et toutes les entrées $x \in \{0, 1\}^n$

$$x \in L \Rightarrow \Pr_r[\mathcal{A}(x, r) = 1] \geq 1/2$$

$$x \notin L \Rightarrow \Pr_r[\mathcal{A}(x, r) = 1] \leq 1/2$$

où la probabilité est prise sur l'aléa interne de l'algorithme.

Les classes *BPP* et *PP* sont très proches au niveau de la définition mais a priori ne sont pas égales. En effet *BPP* est la classe des problèmes qui peuvent être décidés par un algorithme en temps polynomial avec une probabilité de bonne réponse supérieure à une constante elle-même strictement plus grande que $1/2$. *BPP* est donc incluse dans *PP*.

Les classes *BPP* et *BQP* sont comparable en classique et en quantique. Il est connu de $P \subset BPP$. L'autre inclusion est une question ouverte. En terme plus généraux, la question est de savoir si l'aléatoire est utile pour accélérer le calcul ou non. Il y a eu à ce sujet un changement d'avis de la part de la communauté de la complexité : jusqu'aux années 80, la plupart des chercheurs pensaient que *BPP* était différente de *P*, puis divers résultats ont bousculé cette croyance. Aujourd'hui une égalité est souvent envisagée. En plus de contenir les algorithmes qui résolvent les problèmes de décision en temps polynomial, il contient par exemple les méthodes de Monte Carlo.

On peut avoir des machines plus efficaces si nécessaire, autrement dit on peut remplacer $2/3$ par $1 - \varepsilon$ et $1/3$ par ε (pour tout ε petit), en ne changeant pas la classe. Ce renforcement peut être effectué en lançant plusieurs fois la machine de façon indépendante et en faisant un vote.

3.1.3 Notion de difficulté, complet et exposant de complexité

Donnons maintenant des définitions plus générale sur les complexités :

Définition 5. Soit X une classe de complexité. On dit qu'un problème est *X-difficile* (ou *X-hard*) si on peut le ramener à un problème de complexité X par une réduction polynomiale, c'est à dire qu'il existe une fonction $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ qui permet de passer du problème difficile au problème de classe X .

Définition 6. Soit X une classe de complexité. On dit qu'un problème est *X-complet* si il est à la fois X et X -hard.

Par exemple, le problème du voyageur est un problème NP -complet. Si on arrive à montrer que ce problème est P , alors on arrive à montrer que toute la classe NP est dans P

On peut ainsi réaliser les deux graphiques qu'on appelle diagramme d'Euler :

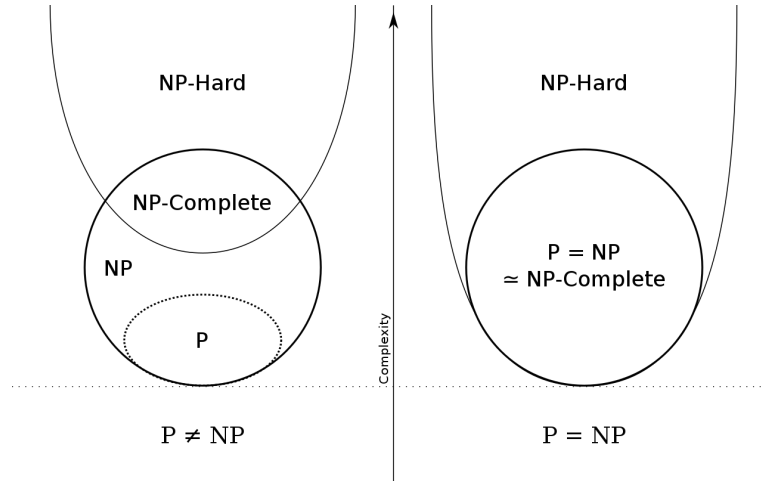


FIGURE 3.1 – Diagramme d'Euler

Définition 7. Soit X et Y deux classes de complexité. On dit qu'un problème est dans la classe X^Y (nous écrivons une classe de complexité X en exposant d'une autre classe Y) si le problème est dans X ayant accès à ce que l'on peut appeler un oracle (boîte noire) ayant accès à un algorithme résolvant des problèmes arbitraires dans la classe Y .

On peut dire que X^Y est l'ensemble des problèmes de décision dans X augmentée d'un oracle pour un problème complet de la classe Y . C'est des classes de complexité qui sont importantes en théorie dans ce que l'on va introduire comme la hiérarchie polynomial de complexité et le théorème de Toda.

3.2 Classe de complexité des problèmes de fonction et $\# P$

3.2.1 Problème de fonction

Un problème de fonction est un problème de calcul où une seule sortie (d'une fonction totale) est attendue pour chaque entrée, mais la sortie est plus complexe que celle d'un problème de décision. Pour les problèmes de fonction, la sortie n'est pas simplement oui ou non.

Ainsi, on peut associer à chaque classe de complexité de décision une classe de complexité de fonction. Il suffit de rajouter un F devant le nom de la classe de décision analogue.

P devient FP , NP devient FNP et BPP devient $FBPP$. Tout comme P et FP sont étroitement liés, NP est étroitement lié à FNP . $FP = FNP$ si et seulement si $P = NP$.

3.2.2 Classe $\#P$

Définition 8. La classe de fonctions $\#P$ est la classe de toutes les fonctions $f : \{0, 1\}^* \rightarrow \mathbb{N}$ pour lesquelles il existe un algorithme classique en temps polynomial \mathcal{A} et un polynôme $p : \mathbb{N} \rightarrow \mathbb{N}$ tel que

$$f(x) = \text{Card}\left(\{y \in \{0, 1\}^{p(|x|)} : \mathcal{A}(x, y) = 1\}\right)$$

$\#P$ est la classe des fonctions qui comptent le nombre de certificats d'un problème de décision qui est dans la classe NP . La classe $\#P$ tient une place à part dans la théorie de la complexité, car ce n'est pas une classe de problèmes de décision mais une classe de fonctions de comptage de solutions.

Un problème de décision de la classe NP est souvent de la forme "Y a-t-il des instances qui satisfont certaines contraintes?". A contrario, les problèmes de la classe $\#P$ correspondants posent la question "Combien y a-t-il d'instances qui satisfont certaines contraintes?".

Clairement, un problème $\#P$ doit être au moins aussi dur que le problème qui lui correspond dans NP , puisque savoir combien il y a de solutions nous dit, en particulier, s'il y a au moins une solution. Ainsi, le problème de la classe $\#P$ correspondant à un problème NP -complet doit être NP -difficile. Certains problèmes de $\#P$ qui sont considérés comme difficiles correspondent à des problèmes faciles, de la classe P . Par exemple le problème du calcul du permanent est $\#P$ -complet alors que le problème d'existence associé est dans P .

De plus, les classes PP et $\#P$ sont fortement liées. Comme ce sont respectivement des classes de décisions et de fonctions, on ne peut pas les comparer directement. Néanmoins, on a l'égalité suivante :

$$P^{\#P} = P^{PP}$$

Pour des problèmes polynomiaux nécessitant une fonction de comptage, on pourrait utiliser à la place un algorithme polynomiale pouvant faire appel à un algorithme de décision probabiliste.

3.3 Hiérarchie polynomiale et théorème associés

3.3.1 Hiérarchie polynomiale

Il n'est pas exagéré de dire que la conjecture que $P \neq NP$ est en effet l'une des, sinon la, plus testée et étudiée affirmation non prouvée dans laquelle les scientifiques de diverses disciplines ont confiance. Cette intuition repose, entre autres, sur l'existence présumée de problèmes dont les solutions sont difficiles à trouver mais faciles à vérifier.

Si $P = NP$, la sécurité de nos comptes bancaires ou autres serait à revoir totalement et nous aurions du mal à trouver des solutions autres que par le quantique. En effet, la possibilité de la cryptographie à clé publique repose sur le fait que $P \neq NP$.

La hiérarchie polynomiale est une hiérarchie de classes de complexité qui étend la notion de classes P, NP . Cette généralisation postule que les niveaux d'une hiérarchie infinie de classes de complexité

sont des sous-ensembles stricts les uns des autres. L'étude d'algorithmes hypothétiques à l'intérieur et à l'extérieur de cette hiérarchie nous permet également de comprendre la complexité informatique de l'approximation des fonctions $\#P$.

On peut prendre cette définition qui se base de l'exposant d'une complexité définie plus haut, qui est la définition avec les algorithmes à oracle (boîte noire auquel on a accès) :

Définition 9 (Hiérarchie polynomiale). *La hiérarchie polynomiale est l'ensemble des classes $(\Sigma_i^P)_{i \in \mathbb{N}}$ tel que :*

$$\begin{aligned}\Sigma_0^P &= P \\ \forall i \in \mathbb{N}^*, \Sigma_{i+1}^P &= NP^{\Sigma_i^P}\end{aligned}$$

On appelle PH l'union des classes de complexité de la hiérarchie polynomiale :

$$PH = \bigcup_{i \in \mathbb{N}} \Sigma_i^P$$

Voici une définition qui définit similairement ces ensembles, à l'aide des quantificateurs \forall et \exists :

Définition 10. *Pour $i \in \mathbb{N}$, un langage $L \subset \{0,1\}^*$ est dans Σ_i s'il existe un polynôme q et une famille de circuits $\{\mathcal{A}_n\}_{n \leq 1}$ à temps polynomial uniforme tels que $x \in L$ si et seulement si $\exists u_1 \in \{0,1\}^k, \forall u_2 \in \{0,1\}^k, \dots, Q_i u_i \in \{0,1\}^k : \mathcal{A}(x, u_1, \dots, u_i) = 1$, où $k = q(|x|)$ et Q_i désigne un quantificateur \forall ou \exists selon que i est pair ou impair, respectivement. La classe PH est l'ensemble $\bigcup_{i \in \mathbb{N}} \Sigma_i$.*

La hiérarchie est intéressante pour la raison suivante : On sait qu'une égalité entre classes d'un même niveau ou de niveaux consécutifs dans la hiérarchie impliquerait un "effondrement" de la hiérarchie à ce niveau.

L'effondrement est le fait que ce qui signifie que si à un niveau i deux classes sont égales, alors toutes les classes "au-dessus" sont égales. On parle alors d'effondrement de la hiérarchie polynomiale au niveau i . On peut le décrire mathématiquement comme cela :

Théoreme 2. *Si $\exists j > i$ tq $\Sigma_i^P = \Sigma_j^P$ alors $\forall j > i, \Sigma_i^P = \Sigma_j^P$ et $\Sigma_i^P = PH$*

Par exemple un exemple au premier niveau effondrement impliquerai que $P = NP$ ce qui est plutôt improbable. C'est en utilisant cet outil qu'on peut montrer que des problèmes sont, sous réserve de la véracité de $P \neq NP$, probablement impossible à résoudre.

La hiérarchie polynomiale est une hiérarchie de classes de complexité définie par l'ajout d'oracles NP consécutifs, où toute couche est supposée contenir strictement toutes les couches inférieures.

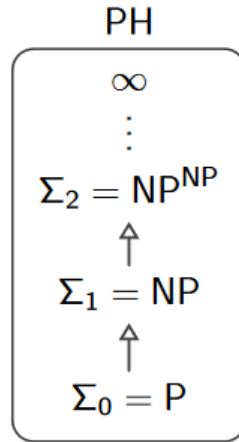


FIGURE 3.2 – Hiérarchie polynomiale

3.3.2 Théorème de Toda

Désormais, il faut faire entre la classe de fonction qui nous intéressera, $\#P$, et la hiérarchie polynomiale qui permet d'exprimer l'infaisabilité à résoudre un problème. On peut trouver la relation entre $\#P$ et PH . Le théorème de Toda nous dit que le calcul exact de $\#P$ fonctions permet de résoudre toute tâche dans PH [6].

Théorème 3 (Théorème de Toda (1991)).

$$PH \subset P^{\#P}$$

On ne peut pas comparer directement une classe de problèmes de décision (PH) à une classe de fonctions ($\#P$). Dans l'énoncé, on utilise $\#P$ en oracle à la classe P . Cela signifie que l'algorithme dans la classe $P^{\#P}$ est polynomiale et doit utiliser une fonction (ou oracle) $f \in \#P$, fonction de comptage.

Le théorème dit, en d'autres termes, que pour tout problème dans la hiérarchie polynomiale, il existe une réduction polynomiale à un problème de comptage.

4 Théorème d'Aaronson & Arkhipov

4.1 Encoder une matrice unitaire dans un circuit linéaire optique

Supposons maintenant que nous ayons une matrice $X \in \mathbb{C}^{nn}$, et que nous souhaitons calculer $|\text{Perm}(X)|^2$. Notez qu'il n'y a aucune perte de #P-complétude ici; c'est déjà un problème #P-complet. Naturellement, comme nous avons le Boson random sampling à notre disposition, nous pourrions construire un réseau de séparateurs de faisceaux qui donne lieu à la matrice X et ainsi simuler le Boson random sampling. Cependant, il y a deux difficultés :

- X n'est pas forcément unitaire
- Avec l'échantillonnage, on ne peut pas calculer exactement, mais seulement une approximation.

Alors, comment "encoder X " dans une matrice unitaire U afin de pouvoir calculer le permanent de la matrice X ? Et pouvons nous être sûr d'avoir une approximation assez bonne et pouvoir contrôler cette précision?

4.2 Encoder une matrice quelconque en matrice unitaire

Pour résoudre le problème d'unitarité de X , nous pouvons construire une U unitaire, qui a un bloc contenant X . Le théorème suivant nous donne l'existence d'une telle matrice mais aussi une méthode pour la construire :

Théoreme 4. Soit $X \in \mathbb{K}^{n \times n}$ tel que $\|X\| \leq 1$. Alors il existe une matrice unitaire $U \in \mathbb{K}^{2n \times 2n}$ tel que :

$$U = \left(\begin{array}{c|c} X & \\ \hline & \end{array} \right)$$

Il faut que $\|X\| \leq 1$. Cela ne pose pas de problème, on peut prendre $Y = \frac{X}{\|X\|}$ et avec $\|Y\| = 1$, nous pouvons réaliser cette construction.

Preuve. Comme $\|X\| \leq 1$, on peut prouver que la matrice $T = I - X^\dagger X$ est semi-définie positive.

$$\begin{aligned} \forall u \in \mathbb{K}^n, \langle u, Tu \rangle &= \langle u, (I - X^\dagger X)u \rangle \\ \langle u, u \rangle - \langle u, X^\dagger Xu \rangle &= \langle u, u \rangle - \langle Xu, Xu \rangle = \|u\|^2 - \|Xu\|^2 \\ &\geq \|u\|^2(1 - \|X\|^2) \geq 0 \end{aligned}$$

Ainsi, comme T est positive semi-positive, d'après la décomposition de Cholesky (pas la factorisation en matrice triangulaire), il existe $Q \in \mathbb{K}^{n \times n}$ tel que $Q^\dagger Q = T = I - X^\dagger X$.

Posons désormais $Y = \begin{pmatrix} X \\ Q \end{pmatrix}$. Donc, $Y^\dagger Y = Q^\dagger Q + X^\dagger X = I$ et les n vecteurs colonnes forment une famille orthonormale de vecteurs dans une dimension $2n$. Par le théorème de la base incomplète, et à l'aide d'un procédé de Gram-Schmidt, nous pouvons compléter Y pour avoir une base orthonormée de $\mathbb{K}^{n \times n}$ et

$$U = \left(\begin{array}{c|c} X & B_1 \\ \hline Q & B_2 \end{array} \right)$$

□

Désormais, nous pouvons prendre n'importe quelle matrice $X \in \mathbb{K}^{n \times n}$ et la normaliser ($X \rightarrow X/\|X\|$) pour avoir $\|X\| = 1$.

On peut trouver une matrice unitaire $U = \left(\begin{array}{c|c} X & \\ \hline & \end{array} \right) \rightarrow \langle I | \phi(U) | T \rangle = |\text{Perm}(X)|^2$

avec $I = |1, \dots, 1, 0, \dots, 0\rangle$

Le modèle du Boson random sampling peut être simulée selon une probabilité dépendant de $|\text{Perm}(X)|^2$ pour n'importe quelle matrice.

4.3 Problème de l'exactitude du calcul du permanent

Ce n'est pas la même chose que d'être capable de calculer explicitement $|\text{Perm}(X)|^2$. En particulier, la permanence, et donc la probabilité, pourrait être exponentiellement petite. Nous savons que nous pouvons obtenir une estimation de $|\text{Perm}(X)|^2$ en réalisant l'expérience plusieurs fois et en calculant la moyenne de tous les résultats des mesures. Mais pour obtenir une estimation non triviale, nous devrions peut-être répéter l'expérience un nombre exponentiel de fois, auquel cas nous pourrions tout aussi bien calculer le permanent à l'aide d'un ordinateur classique.

Il faut voir maintenant s'il est difficile d'approximer le permanent d'une matrice. Nous allons démontrer ce théorème :

Théoreme 5. *Soit $X \in \mathbb{K}^{n \times n}$ tel que $\|X\| \leq 1$. Alors il existe un algorithme classique randomisé qui approxime $|\text{Perm}(X)|$ à une erreur additive $\pm \varepsilon$ en $O(\frac{n^2}{\varepsilon^2})$*

Pour prouver cela, nous avons besoin d'utiliser une autre formulation du permanent introduite par Glynn. Cette formule permet de réaliser l'algorithme classique le plus rapide connue aujourd'hui pour calculer le permanent exactement. Il est d'une complexité $O(2^n n^2)$, et peut être réduit en complexité $O(2^n n)$ en itérant les $z_1 \dots z_n$ selon le code de Gray.

Lemme 3 (Formule de Glynn). *Pour toute matrice $X \in \mathbb{K}^{n \times n}$,*

$$\text{Perm}(X) = \mathbb{E}_{z_1, \dots, z_n \in \{1, -1\}} \left[z_1 \dots z_n \prod_{i=1}^n (x_{i1} z_1 + \dots + x_{in} z_n) \right]$$

Le terme à l'intérieur de l'espérance s'appelle le terme de Ryser et est définie comme ceci :

$$\forall z = (z_i)_{i \in \{1, \dots, n\}} \in \{-1, 1\}^n$$

$$\text{Rys}_z(X) = z_1 \dots z_n \prod_{i=1}^n (x_{i1} z_1 + \dots + x_{in} z_n)$$

et

$$\text{Perm}(X) = \mathbb{E}_{z_1, \dots, z_n \in \{1, -1\}} [\text{Rys}_z(X)]$$

C'est un estimateur non biaisé pour le permanent.

Preuve. On peut réécrire ce terme (qu'on appellera $J(X)$) comme ceci en moyennant sur tout les $(z_i)_{i \in \{1, \dots, n\}}$:

$$J(X) = \frac{1}{2^n} \sum_{(z_i)_{i \in \{-1, 1\}^n}} \left[\left(\prod_{j=1}^n z_j \right) \prod_{i=1}^n \sum_{k=1}^n x_{ik} z_k \right]$$

On peut utiliser maintenant la formule pour intervertir produit et somme (en rappelant que $\mathcal{F}_{m,n} = \{f, f : \llbracket 1, m \rrbracket \rightarrow \llbracket 1, n \rrbracket\}$:

$$\begin{aligned} J(X) &= \frac{1}{2^n} \sum_{(z_i)_{i \in \{-1, 1\}^n}} \left[\left(\prod_{j=1}^n z_j \right) \sum_{f \in \mathcal{F}_{n,n}} \prod_{i=1}^n x_{i, f(i)} z_{f(i)} \right] \\ &= \sum_{f \in \mathcal{F}_{n,n}} \frac{1}{2^n} \sum_{(z_i)_{i \in \{-1, 1\}^n}} \left[\left(\prod_{j=1}^n z_j \right) \left(\prod_{i=1}^n z_{f(i)} \right) \left(\prod_{i=1}^n x_{i, f(i)} \right) \right] \\ &= \sum_{f \in \mathcal{F}_{n,n}} \left(\frac{1}{2^n} \sum_{(z_i)_{i \in \{-1, 1\}^n}} \prod_{i=1}^n z_i z_{f(i)} \right) \left(\prod_{i=1}^n x_{i, f(i)} \right) \\ &= \sum_{f \in \mathcal{F}_{n,n}} \eta(f) \left(\prod_{i=1}^n x_{i, f(i)} \right) \end{aligned}$$

avec $\eta(f) = \left(\frac{1}{2^n} \sum_{(z_i)_{i \in \{-1, 1\}^n}} \prod_{i=1}^n z_i z_{f(i)} \right)$

Montrons alors le lemme suivant :

Lemme 4. Soit $f : \llbracket 1, n \rrbracket \rightarrow \llbracket 1, n \rrbracket$ et définissons :

$$\eta(f) = \left(\frac{1}{2^n} \sum_{(z_i)_{i \in \{1, \dots, n\}}} \prod_{i=1}^n z_i z_{f(i)} \right)$$

Alors, il n'y a que 2 possibilités de résultat pour η :

$$\eta(f) = \begin{cases} 1 & \text{si } f \in S_n \\ 0 & \text{sinon.} \end{cases}$$

Preuve. Ce terme s'annule quand f n'est pas une permutation car dès que f n'est pas une injection, il y a une "redondance" dans le produit de $z_i z_{f(i)}$ qui s'annule avec la somme sur les $(z_i)_i \in \{-1, 1\}^n$.

Sinon, si f est injective, elle est donc bijective et c'est une permutation. Alors $\prod_{i=1}^n z_i z_{f(i)} = 1$ car tout les termes se retrouve au carré et $\sum_{(z_i)_i \in \{-1, 1\}^n} (\prod_{i=1}^n z_i z_{f(i)}) = 2^n$ donc $\eta(f) = 1$. \square

Ainsi, la somme sur $\mathcal{F}_{n,n}$ est au final une somme sur S_n et :

$$J(X) = \sum_{f \in S_n} \left(\prod_{i=1}^n x_{i, f(i)} \right) = \text{Perm}(X)$$

\square

En effet, quand on développe le produit, tous les termes où les termes z_i qui sont élevés au carré ou plus s'annule avec le produit $z_1 \dots z_n$. Il reste $n!$ termes qui correspondent aux termes du permanent. A partir de la formule de Glynn, on peut montrer facilement une propriété intéressante du permanent :

Théoreme 6. $|\text{Rys}_z(X)| \leq \|X\|^n$ et dans le cas particulier d'une matrice unitaire U , $|\text{Rys}_z(U)| \leq 1$

Preuve. On va utiliser la formule de Glynn mais aussi l'inégalité entre moyenne géométrique et arithmétique :

$$\left(\prod_{i=1}^n a_i \right)^{\frac{1}{n}} \leq \frac{1}{n} \sum_{i=1}^n a_i$$

\square

et en utilisant la formule de Glynn :

$$\begin{aligned} \sqrt{\prod_{i=1}^n |x_{i1}z_1 + \dots + x_{in}z_n|^2} &\leq \left(\frac{\sqrt{\sum_{i=1}^n |x_{i1}z_1 + \dots + x_{in}z_n|^2}}{\sqrt{n}} \right)^n \\ &= \left(\frac{\|Xz\|}{\|z\|} \right)^n \end{aligned}$$

$$\leq \|X\|^n$$

et nous obtenons : $|\text{Rys}_z(X)| \leq \|X\|^n$

Cela implique le corolaire ci dessous :

Corrolaire 1. $|\text{Perm}(X)| \leq \|X\|^n$ et dans le cas particulier d'une matrice unitaire U , $|\text{Perm}(U)| \leq 1$

Preuve.

$$|\text{Perm}(X)| = \left| \mathbb{E}_{(z_i)_{i \in \{1, -1\}^n}} (\text{Rys}_z(X)) \right| \leq \mathbb{E}_{(z_i)_{i \in \{1, -1\}^n}} (|\text{Rys}_z(X)|) \leq \|X\|^n$$

□

Maintenant que nous avons cela, nous avons un algorithme efficace qui estime une approximation du permanent de X . Cette propriété nous permet d'assurer l'approximation de l'algorithme d'estimation du permanent. L'algorithme peut être décrit comme ci dessous :

Algorithme 1 : Algorithme d'approximation du permanent.

Résultat : Approximer le permanent avec une erreur ε

Choisir aléatoirement $\frac{1}{\varepsilon^2}$ vecteurs $Z = (z_1, \dots, z_n) \in \{1, -1\}^n$. On nommera cette ensemble V_ε ;

Calculer $\sum_{z \in V_\varepsilon} z_1 \dots z_n \prod_{i=1}^n (x_{i1}z_1 + \dots + x_{in}z_n)$;

Retourner ce résultat qui correspond à une moyenne empirique.

On approxime $|\text{Perm}(X)|$ avec une erreur ε en $O(\frac{n^2}{\varepsilon^2})$.

En effet, la probabilité d'échec, avec $T = O(\frac{1}{\varepsilon^2})$, en choisissant $z(1), \dots, z(T)$ des vecteurs de $\{-1, 1\}^n$, d'après l'inégalité de Chernoff :

$$\begin{aligned} Pr \left[\left| \text{Perm}(X) - \frac{1}{T} \sum_{t=1}^T \text{Rys}_{z(t)}(X) \right| \geq \varepsilon \|X\|^n \right] &\leq 2 \exp\left(-\frac{2T(\varepsilon \|X\|^n)^2}{(2\|X\|^n)^2}\right) \\ &\leq 2 \exp\left(-\frac{\varepsilon^2 T}{2}\right) \end{aligned}$$

$$\iff Pr \left[\left| \text{Perm}(X) - \frac{1}{T} \sum_{t=1}^T \text{Rys}_{z(t)}(X) \right| \leq \varepsilon \|X\|^n \right] \geq 1 - 2 \exp\left(-\frac{\varepsilon^2 T}{2}\right)$$

En ayant choisit $T = O(\frac{1}{\varepsilon^2})$, on s'assure de l'approximation du permanent d'un coefficient additif $\varepsilon \|X\|^n$ par de l'estimateur de Ryster.

Si X est unitaire, alors on s'assure de l'approximation du permanent d'un coefficient additif ε par de l'estimateur de Ryster. $\text{Rys}_z(X)$ peut être calculer en $O(n^2)$ étapes.

Ainsi, on a un algorithme qui permet de calculer une approximation du permanent en $O(n^2/\varepsilon^2)$ étapes.

Alors, avons-nous démontré que les réseaux de séparateurs de faisceaux ne présentent finalement aucun avantage par rapport aux ordinateurs classiques ? C'est là que BosonSampling entre en jeu. Nous pouvons revoir l'expérience et observer qu'elle échantillonne une certaine distribution de probabilité sur un nombre exponentiel de résultats possibles. Nous savons que cette distribution connaît en quelque sorte "implicitement" les permanents. Supposons donc que notre problème consiste à échantillonner à partir de cette distribution de probabilité basée sur les permanents ; alors peut-être qu'un ordinateur quantique peut donner un avantage.

4.4 Théorème d'Aaronson

Dans cette partie, nous allons expliquer pourquoi le Boson random sampling est un problème qui pourrait montrer la suprématie quantique. Introduisons tout d'abord les complexités algorithmiques que nous aurons à traiter.

4.4.1 Complexité du calcul du permanent

Le permanent et le déterminant sont deux valeurs que l'on peut associer à une matrice qui sont quasi semblable en terme de définition :

$$\text{Perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i, \sigma_i}$$

$$\det(A) = \sum_{\sigma \in S_n} \varepsilon(\sigma) \prod_{i=1}^n a_{i, \sigma_i}$$

Le terme multiplicatif qui change est la signature de la permutation :

$$\varepsilon(\sigma) = \prod_{1 \leq i < j \leq n} \text{sgn}(\sigma(j) - \sigma(i))$$

S'il existe une formule générale de calcul du déterminant, calculer directement à partir de la formule serait une technique difficile à mettre en œuvre pour des matrices de grande taille. On lui préfère alors des méthodes de calcul plus simples comme la technique du pivot de Gauss. La méthode de Gauss, pour inverser une matrice carrée de m lignes et m colonnes (ou pour calculer son déterminant) possède une complexité $O(m^3)$, autrement dit $O(n^{3/2})$, si $n = m^2$ désigne la taille de la matrice. Donc le problème du calcul du déterminant est dans FP .

La seule chose qui change entre le déterminant et le permanent est la signature, coefficient multiplicatif devant le produit des termes de la matrice. Ces signatures $\varepsilon(\sigma) \in \{-1, 1\}$ change totalement la complexité des algorithmes de calculs.

En effet, le permanent est beaucoup plus difficile à calculer que le déterminant. Il n'existe pas d'analogue du pivot de Gauss pour les permanents. Plus précisément, le problème du calcul du permanent de matrice 0-1 peut être vu comme un problème de comptage. En effet, il faut calculer tous les produits de termes de la matrice complexe pour chaque permutation.

En 1971, Valiant trouve la complexité du calcul du permanent :

Théoreme 7 (1971 Valiant). *Le calcul du permanent d'une matrice $0-1$ est un problème $\#P$ -complet.*

C'est à partir de ce résultat de complexité que Aaronson et Arkhipov ont vu que le lien entre le Boson random sampling et la complexité de calcul pouvait être intéressant.

Il faut alors généraliser le problème avec une matrice quelconque $X \in \mathbb{K}^{n \times n}$. Aaronson a décrit la complexité d'approximer $|\text{Perm}(X)|^2$ d'un coefficient multiplicatif g :

Théoreme 8. *L'approximation du permanent $|\text{Perm}(X)|^2$ est un problème $\#P$ -difficile.*

La raison pour laquelle nous nous intéressons à $|\text{Perm}(X)|^2$ plutôt qu'à $\text{Perm}(X)$ lui-même est que les probabilités de mesure dans le Boson random sampling sont les carrés des modules des permanents.

Preuve. On sait que calculer exactement un permanent est $\#P$ -complet. Nous devons montrer comment calculer $\text{Perm}(X)$ exactement, étant donné un oracle qui fait une approximation multiplicative des permanents.

Soit \mathcal{O} un oracle qui, étant donné une matrice $M \in \mathbb{K}^{n \times n}$ sort un nombre $\mathcal{O}(M)$ tel que

$$\frac{|\text{Perm}(M)|^2}{g} \leq \mathcal{O}(M) \leq g |\text{Perm}(M)|^2$$

De plus, prenons $X = (x_{ij}) \in \{0, 1\}^{n \times n}$ être une matrice d'entrée, que nous supposons pour simplifier être constituée uniquement de 0 et de 1. Ensuite, nous montrerons comment calculer $\text{Perm}(X)$ exactement, en temps polynomial et en utilisant $O(gn^2 \log n)$ requêtes adaptatives à \mathcal{O} . Puisque $\text{Perm}(X)$ est $\#P$ -complet par le théorème de Valiant, cela impliquera immédiatement le théorème ci dessus comme quoi l'approximation est un problème plus difficile qu'un problème $\#P$ auquel on se ramène en temps polynomial. Puisque X est positive, nous pouvons vérifier en temps polynomial si $\text{Perm}(X) = 0$. Si $\text{Perm}(X) = 0$, nous avons terminé, alors supposons que $\text{Perm}(X) \geq 1$. Alors il existe une permutation σ telle que $x_{1,\sigma(1)} = \dots = x_{n,\sigma(n)} = 1$. De plus, nous pouvons trouver une telle permutation σ en temps polynomial ; en effet, ceci est équivalent au problème standard de trouver un match parfait dans un graphe biparti. En permutant les lignes et les colonnes, nous pouvons supposer sans perte de généralisation du théorème que $x_{11} = \dots = x_{nn} = 1$.

Prenons :

$$X = \begin{pmatrix} x_{11} & \\ & [Y] \end{pmatrix}$$

$$X^{[r]} = \begin{pmatrix} x_{11} - r & \\ & [Y] \end{pmatrix}$$

Ici, X est une matrice $n \times n$ et Y est la sous-matrice en bas à droite de taille $(n-1) \times (n-1)$ de la matrice X . $X^{[r]}$ est la matrice X avec le nombre r qui soustrait le nombre en haut à droite. On a alors :

$$\text{Perm}(X^{[r]}) = \text{Perm}(X) - r \text{Perm}(Y)$$

Nous supposons alors par induction que nous connaissons déjà $\text{Perm}(Y)$. Nous utiliserons ceci, ainsi que $O(gn \log n)$ requêtes à \mathcal{O} pour trouver $\text{Perm}(X)$.

On cherche alors un certain r à l'aide d'un algorithme de recherche dichotomique tel que :

$$\begin{aligned} \text{Perm}(X^{[r]}) &= 0 \\ \implies \text{Perm}(X) - r \text{Perm}(Y) &= 0 \\ \implies \text{Perm}(X) &= r \text{Perm}(Y) \end{aligned}$$

Comme $\text{Perm}(Y) \geq 1$. Il existe alors une unique valeur r^* tel que $\text{Perm}(X) = r^* \text{Perm}(Y)$ et :

$$r^* = \frac{\text{Perm}(X)}{\text{Perm}(Y)}$$

Prenons $r_0 = 0$. On va chercher de manière itérative, r_1, r_2, \dots . Nous souhaitons maintenir :

$$\mathcal{O}(X^{[r_{t+1}]}) \leq \frac{\mathcal{O}(X^{[r_t]})}{2}$$

Pour trouver r_{t+1} à partir de r_t , on remarque tout d'abord en utilisant l'approximation multiplicative que fait \mathcal{O} :

$$\begin{aligned} |r_t - r^*| &= \frac{|r_t \text{Perm}(Y) - \text{Perm}(X)|}{|\text{Perm}(Y)|} \\ &= \frac{|\text{Perm}(X^{[r_t]})|}{|\text{Perm}(Y)|} \\ &\leq \frac{\sqrt{g \mathcal{O}(X^{[r_t]})}}{|\text{Perm}(Y)|} \end{aligned}$$

Posons alors :

$$\beta := \frac{\sqrt{g \mathcal{O}(X^{[r_t]})}}{|\text{Perm}(Y)|}$$

Alors, on sait maintenant que $r^* \in I := [r_t - \beta, r_t + \beta]$. On peut maintenant diviser I en L segment de même longueur et on définit $(s_i)_{i \in \{1, L\}}$ les bords à gauche de chaque segment. Le but maintenant est d'évaluer $\mathcal{O}(X^{[s_i]})$ pour chaque i allant de 1 à L et on pourra prendre $r_{t+1} = s_j$ avec j qui minimise $\mathcal{O}(X^{[s_j]})$. Clairement, on sait qu'on aura un j tel que $|s_j - r^*| \leq \beta/L$, et on aura pour ce choix particulier :

$$\mathcal{O}(X^{[s_j]}) \leq g \text{Perm}(X^{[s_j]})^2$$

$$\begin{aligned}
&= g(\text{Perm}(X) - s_j \text{Perm}(Y))^2 \\
&= g(\text{Perm}(X) - (s_j - r^*) \text{Perm}(Y) - r^* \text{Perm}(Y))^2 \\
&= g(s_j - r^*)^2 \text{Perm}(Y)^2 \\
&\leq g \frac{\beta^2}{L^2} \text{Perm}(Y)^2 \\
&= \frac{g^2}{L^2} \mathcal{O}(X^{[r_t]})
\end{aligned}$$

Pour le passage de la ligne 3 à 4, on utilise le fait que $\text{Perm}(X) = r^* \text{Perm}(Y)$.

Si on prend $L \leq \sqrt{2}g$ on obtient :

$$\mathcal{O}(X^{[r_{t+1}]}) \leq \mathcal{O}(X^{[s_j]}) \leq \frac{\mathcal{O}(X^{[r_t]})}{2}$$

et c'est ce que nous souhaitons pour la recherche dichotomique.

Enonçons maintenant un théorème énoncé par Brègman :

Théoreme 9 (Brègman (1973)). *Soit X une 0 – 1 matrice. Alors :*

$$\text{Perm}(X) \leq \prod_{i=1}^n (\xi_i!)^{1/\xi_i} \leq n!$$

avec

$$\xi_i := \sum_{j=1}^n x_{ij}$$

En regardant l'estimation initiale :

$$\mathcal{O}(X^{[r_0]}) = \mathcal{O}(X) \leq g \text{Perm}(X)^2 \leq g(n!)^2$$

Pour un $T = O(n \log n)$

$$\mathcal{O}(X^{[r_T]}) \leq \frac{\mathcal{O}(X^{[r_0]})}{2^T} \leq \frac{g(n!)^2}{2^T} << \frac{1}{4g}$$

Cela implique le fait que :

$$|r_T - r^*| \leq \frac{g \mathcal{O}(X^{[r_T]})}{\text{Perm}(Y)} << \frac{1}{2 \text{Perm}(Y)}$$

Mais cela signifie que nous pouvons trouver r^* exactement, puisque r^* est égal à un nombre rationnel $\text{Perm}(X)/\text{Perm}(Y)$ où $\text{Perm}(X)$ et $\text{Perm}(Y)$ sont tous deux des entiers positifs et $\text{Perm}(Y)$ est connu. On converge alors vers ce r^* pour la première réduction de taille de la matrice qu'on appellera $r_{[1]}$

Ainsi, on peut réduire le problème le problème de calcul de $\text{Perm}(X)$ au calcul de $\text{Perm}(Y)$, où Y est une matrice de taille $(n-1) \times (n-1)$. On peut alors réitérer cette réduction jusqu'à qu'on est :

$$\text{Perm}(X) = \prod_{k=1}^n r_{[k]}$$

On se ramène donc en temps polynomial à un problème $\#P$ -complet : le problème d'approximation est donc un problème $\#P$ -hard et on a un algorithme qui nous permet de faire cela.

□

4.4.2 Échantillonnage exact du Boson random sampling

Maintenant que nous avons énoncé la complexité du calcul de $|\text{Perm}(X)|^2$, nous pouvons énoncer le théorème centrale qui a poussé l'intérêt du problème Boson random sampling :

Théoreme 10 (Aaronson & Arkhipov (2011)). *Le problème exact du Boson Random Sampling n'est pas efficacement solvable par un ordinateur classique, à moins que $P^{\#P} = BPP^{NP}$ et que la hiérarchie polynomiale s'effondre au troisième niveau.*

Ce théorème semble compliqué mais il nous dit que si il existe un algorithme classique qui puisse simuler l'échantillonnage aléatoire de Boson, alors c'est comme si $P = NP$. Le problème $P = NP$ est un des quelques problèmes dit du "millénaire" qui n'a toujours pas été résolu. Néanmoins, les spécialistes de la complexité conjecture que P serait différent de NP . Ainsi, sous réserve de la véracité de $P \neq NP$, il ne serait pas possible de simuler à l'aide d'un ordinateur classique le Boson random sampling .

Preuve. Pour prouver ce théorème, nous avons besoin d'énoncer un théorème prouvé en 1983 par Stockmayer, qui nous donne l'existence d'algorithme d'approximation d'une fonction booléenne :

Théoreme 11 (1983 Stockmeyers). *Soit une fonction $f : \{0, 1\}^n \rightarrow \{0, 1\}$ booléenne et*

$$p = \Pr_{x \in \{0,1\}^n} [f(x) = 1] = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)$$

.

Alors, $\forall c \leq 1 + 1/\text{poly}(n)$, il existe un algorithme de classe $FBPP^{NP}$ qui approxime p avec une erreur multiplicative c

Intuitivement, le théorème dit qu'une machine BPP^{NP} peut toujours estimer la probabilité p qu'un algorithme randomisé en temps polynomial accepte le résultat à un facteur multiplicatif $1/\text{poly}(n)$ près, même si p est exponentiellement petit.

Une autre interprétation du théorème est que tout problème de comptage qui implique l'estimation de la somme de 2^n nombres réels positifs peut être résolu approximativement avec un algorithme de

complexité $BPP^N P$.

En revanche, si un problème de comptage implique l'estimation d'une somme de nombres positifs et négatifs-par exemple, si l'on voulait approximer $\mathbb{E}_{x \in \{0,1\}^n} (f(x))$ pour une certaine fonction $f : \{0,1\}^n \rightarrow \{-1,1\}$, alors la situation est complètement différente. On a pu montrer auparavant que même l'approximation multiplicative est $\#P$ -dure, et donc peu susceptible d'être dans $FBPP^{NP}$.

On rappelle qu'un estimateur s fournit une approximation de S d'un facteur multiplicatif $0 < c \leq 1$ si :

$$cS \leq s \leq S/c$$

$$1 - c \leq \left| \frac{S - s}{S} \right| \leq \frac{1 - c}{c}$$

Soit $X \in \mathbb{R}^{n \times n}$ et un paramètre $g \in \left[1 + \frac{1}{poly(n)}, poly(n)\right]$.

On sait qu'approximer le permanent de X avec un facteur multiplicatif g est un problème $\#P$ -difficile. Il faudrait alors montrer qu'approximer $|\text{Perm}(X)|^2$ est un problème $FBPP^{NP}$.

D'après le théorème d'encodage des matrices dans une matrice unitaire, en posant $Y := \varepsilon X$ avec $\varepsilon = 1/\|X\| \geq 2^{poly(n)}$, $\exists U \in \mathbb{K}^{2n \times 2n}$ tel que : $U = \left(\begin{array}{c|c} Y & \\ \hline Q & \end{array} \right)$

Prenons la matrice $A = \left(\begin{array}{c} Y \\ Q \end{array} \right)$ qui correspond aux n premières colonnes de U . Alors, nous nous intéressons à la probabilité p_A d'avoir la sortie $|1_n\rangle = |1 \dots 1 0 \dots 0\rangle$ pour l'entrée X dans un algorithme randomisé \mathcal{O} réalisant le Boson random sampling .

$$\begin{aligned} p_A &= \Pr_r[\mathcal{O}(X, r) = 1_n] \\ &= |\langle 1_n | \varphi(U) | 1_n \rangle| \\ &= |\text{Perm}(U_{n,n})|^2 = |\text{Perm}(Y)|^2 \\ &= \varepsilon^{2n} |\text{Perm}(X)|^2 \end{aligned}$$

Alors, approximer p_a avec un facteur multiplicatif g est un problème $\#P$ -difficile.

D'après le théorème de Stockmeyer, il est possible d'approximer p_A d'un facteur multiplicatif g à l'aide d'un algorithme de classe $FBPP^{NP^O}$.

Il suffit alors maintenant de prouver le corollaire suivant :

Corollaire 2. *Supposons que le Boson random sampling exact puisse être réalisé en temps polynomial avec un algorithme classique. Alors $P^{\#P} = BPP^{NP}$ et donc la hiérarchie polynomiale s'effondre au troisième niveau.*

Preuve. Désormais, il faut montrer que par conséquent, $P^{\#P} = BPP^{NP}$.

On a prouvé que s'il existe un algorithme classique qui résout le problème du Boson random sampling, comme c'est un problème $\#P$ -difficile qui peut être réduit à $FBPP^{NP}$, alors tous les algorithmes de $P^{\#P}$ sont nécessairement dans BPP^{NP} , nous avons $P^{\#P} \subset BPP^{NP}$.

Pour lier BPP avec $\#P$, nous avons ce théorème :

Théorème 12 ((1983) Sipser, Gacs, Lautemann).

$$BPP \subset \Sigma_2^P$$

BPP est donc dans la hiérarchie polynomiale. Comme $BPP \subset \Sigma_2^P = NP^{NP}$, $BPP^{NP} \subset \Sigma_3^P = NP^{NP^{NP}}$. Le théorème de Stockmayer est lié au troisième niveau de la hiérarchie polynomiale.

D'après le théorème de Toda, $PH \in P^{\#P}$, et on a $BPP^{NP} \subset \Sigma_P^3 \subset PH \subset P^{\#P}$. Donc, par double inclusion, $BPP^{NP} = \Sigma_P^3 = PH = P^{\#P}$.

Approximer le carré du permanent d'une matrice est un problème $\#P$ -difficile. Donc, $P^{\#P} = BPP^{NP}$, la hiérarchie polynomiale s'effondre donc au troisième niveau et $BPP^{NP} = \Sigma_P^3$. \square

Soulignons à nouveau qu'un algorithme classique ne devrait être en mesure d'effectuer une approximation multiplicative des probabilités d'acceptation, car la classe Σ_P^3 n'est pas censée être contenue dans BPP . \square

On a ainsi donné la première preuve théorique que l'échantillonnage de Boson est difficile pour les algorithmes classiques. En effet, nous avons montré que s'il existait un algorithme classique efficace pour l'échantillonnage de Boson exact, la hiérarchie polynomiale s'effondrerait. La preuve a été apportée en combinant les deux faits suivants.

- Approximer de manière multiplicative la valeur de $|\text{Perm}(X)|^2$ est $\#P$ -hard.
- S'il existait un algorithme classique efficace pour le Boson random sampling exact, alors $|\text{Perm}(X)|^2$ serait approximable dans BPP^{NP} .

Pour résumer, s'il existe un algorithme efficace qui échantillonne le Boson random sampling :

$$\begin{aligned} \implies & \text{ Les problèmes d'approximation } \#P - \text{difficile sont dans } BPP^{NP} \\ \implies & P^{\#P} \subset BPP^{NP} \\ \implies & PH = BPP^{NP} \\ \implies & PH \text{ s'effondre à } \Sigma_3^P = NP^{NP^{NP}} \end{aligned}$$

Et ce serait un gros problème, car cela signifierait que la hiérarchie polynomiale est finie ! C'est pourquoi nous considérons qu'il est improbable qu'il existe un algorithme aléatoire rapide pour l'échantillonnage de Boson, du moins dans le cadre exact.

Bibliographie

- [1] Scott AARONSON et Alex ARKHIPOV. « The Computational Complexity of Linear Optics ». In : *Theory of Computing* 9 (fév. 2013). Number : 4 Publisher : Theory of Computing, p. 143-252. DOI : 10.4086/toc.2013.v009a004. URL : <https://theoryofcomputing.org/articles/v009a004/> (visité le 12/11/2022).
- [2] L. CHAKHMAKHCHYAN, N. J. CERF et R. GARCIA-PATRON. « A quantum-inspired algorithm for estimating the permanent of positive semidefinite matrices ». In : *Physical Review A* 96.2 (août 2017). arXiv :1609.02416 [quant-ph], p. 022329. ISSN : 2469-9926, 2469-9934. DOI : 10.1103/PhysRevA.96.022329. URL : <http://arxiv.org/abs/1609.02416> (visité le 15/01/2023).
- [3] Bryan T. GARD et al. « An introduction to boson-sampling ». In : arXiv :1406.6767 [quant-ph]. Août 2015, p. 167-192. DOI : 10.1142/9789814678704_0008. URL : <http://arxiv.org/abs/1406.6767> (visité le 08/12/2022).
- [4] Dominik HANGLEITER et Jens EISERT. *Computational advantage of quantum random sampling*. arXiv :2206.04079 [cond-mat, physics :quant-ph]. Nov. 2022. DOI : 10.48550/arXiv.2206.04079. URL : <http://arxiv.org/abs/2206.04079> (visité le 10/11/2022).
- [5] Stefan SCHEEL. *Permanents in linear optical networks*. arXiv :quant-ph/0406127. Juin 2004. DOI : 10.48550/arXiv.quant-ph/0406127. URL : <http://arxiv.org/abs/quant-ph/0406127> (visité le 23/11/2022).
- [6] Lance FORTNOW. « A Simple Proof of Toda's Theorem ». en. In : *THEORY OF COMPUTING* 5 (2009).