

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

5 mars 2023

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Boson Random Sampling</b>	<b>1</b>
2.1	Espace de Fock , opérateur de création et d'annihilation . . . . .	1
2.1.1	Etat de Fock . . . . .	2
2.1.2	Opérateur créateur et annihilateur pour les bosons . . . . .	2
2.2	Element d'optique quantique . . . . .	2
2.3	Boson random sampling . . . . .	3
2.4	Analogie avec la planche de Galton . . . . .	4
2.5	Probabilité de sortie : apparition du permanent . . . . .	4
2.5.1	Positionnement du problème . . . . .	5
2.5.2	Pourquoi le permanent ? Exemple avec 2 photons . . . . .	6
2.5.3	Cas général . . . . .	7
2.6	Analogie en Théorie des graphes . . . . .	8
<b>3</b>	<b>Complexité algorithmique</b>	<b>10</b>
3.1	Classes de complexité des problèmes de décisions . . . . .	10
3.1.1	P et NP . . . . .	10
3.1.2	PP ,BPP, BQP . . . . .	11
3.1.3	Notion de difficulté, complet et exposant de complexité . . . . .	12
3.2	Classe de complexité des problèmes de fonction et $\# P$ . . . . .	12
3.2.1	Problème de fonction . . . . .	12
3.2.2	Classe $\# P$ . . . . .	13
3.3	Hierarchie polynomiale et théorème associés . . . . .	13
3.3.1	Hierarchie polynomiale . . . . .	13
3.3.2	Théorème de Toda . . . . .	14
<b>4</b>	<b>Théorème d'Aaronson &amp; Arkhipov</b>	<b>15</b>
4.1	Encoder une matrice unitaire dans un circuit linéaire optique . . . . .	15
4.2	Encoder une matrice quelconque en matrice unitaire . . . . .	15
4.3	Problème de l'exactitude du calcul du permanent . . . . .	16
4.4	Théorème d'Aaronson . . . . .	19
4.4.1	Complexité du calcul du permanent . . . . .	19

# 1 Introduction

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 qu'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. Le but de ce mémoire est de se familiariser avec cette problématique dans le contexte du quantum random sampling et/ou du machine learning quantique.

Après s'être familiarisée avec la problématique, l'étudiant adressera quelques-unes des questions mathématiques qui se présentent naturellement dans ce contexte. Le stage vient en complément naturel au cours sur l'information quantique du S4.

Le sujet du stage est typiquement un sujet de physique mathématique, et se situe à l'intersection de l'analyse fonctionnelle, des probabilités et de la modélisation. Comme plusieurs questions restent ouvertes, des simulations numériques sont intéressantes et peuvent faire partie du stage, selon les goûts de l'étudiant.

# 2 Boson Random Sampling

Le Boson random sampling est une variante de l'algorithme de quantum computing qui utilise des photons (particules de lumière) pour effectuer des calculs. Cet algorithme a été introduit pour la première fois en 2011 par Scott Aaronson et Alex Arkhipov [1], qui ont montré qu'il était possible d'utiliser un réseau de photons pour effectuer des calculs qui sont difficiles à effectuer de manière classique, même avec des ordinateurs quantiques.

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 irréalisables 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 la supériorité 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.

En résumé, le Boson sampling est un algorithme de quantum computing qui utilise des photons pour effectuer des calculs de manière plus efficace que de manière classique, mais qui est également très difficile à mettre en oeuvre de manière pratique.

## 2.1 Espace de Fock, opérateur de création et d'annihilation

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. De plus, selon l'intégralité du spin des particules, les produits tensoriels doivent être des produits alternatifs (antisymétriques) ou symétriques de l'espace de Hilbert sous-jacent à une particule. Plus précisément :

- Les fermions, possédant un spin demi-entier et obéissant au principe d'exclusion de Pauli, correspondent à des produits tensoriels antisymétriques.
- Les bosons, possédant un spin entier (et non régis par le principe d'exclusion) correspondent à des produits tensoriels symétriques.

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. C'est le champs de l'optique quantique.

### 2.1.1 Etat de Fock

Un état de Fock s'écrit alors comme ceci : donnons  $(s_i)_{i \in I}$  une base orthonormal d'état dans un espace de Hilbert sous-jacent à 1 particule.

Cela induit une base correspondante 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  $N_{s_i}$  correspondant au  $i$ ème élément de l'état de Fock représentant l'état  $s_i$ .

Un état quantique dans l'espace de Fock peut s'écrire comme ceci : les  $m_{s_i} \in \mathbb{N}$  représente le nombre de particule dans l'état  $s_i$ , et on écrit cet état :  $|m_{s_1} \dots m_{s_n}\rangle$ .

Ainsi, l'opérateur  $N_{s_i}$  agit comme ceci sur l'état de Fock :

$$N_{s_i} |m_{s_1} \dots m_{s_n}\rangle = m_{s_i} |m_{s_1} \dots m_{s_n}\rangle$$

### 2.1.2 Opérateur créateur et annihilateur 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,  $\forall |m_{s_1} \dots m_{s_n}\rangle$ , état de l'espace de Fock,

$$\begin{aligned} a_{s_i} |m_{s_1} \dots m_{s_i} \dots m_{s_n}\rangle &= \sqrt{m_{s_i}} |m_{s_1} \dots m_{s_i} - 1 \dots m_{s_n}\rangle \\ a_{s_i}^\dagger |m_{s_1} \dots m_{s_i} \dots m_{s_n}\rangle &= \sqrt{m_{s_i} + 1} |m_{s_1} \dots m_{s_i} + 1 \dots m_{s_n}\rangle \end{aligned}$$

Ce ne sont pas des observables et on peut définir l'opérateur nombre de particule :

$$N_{s_i} = a_{s_i}^\dagger a_{s_i}$$

Les relations de commutation des opérateurs de création et d'annihilation dans un système à bosons multiples sont :

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

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

## 2.2 Element d'optique quantique

En général, pour réaliser un ordinateur quantique, nous avons besoin d'un moyen de préparer des états quantiques, d'effectuer un ensemble de transformation sur les qubits grâce à des portes quantiques universelles et de mesurer l'état de sortie. Pour générer un état quantique, nous utilisons une source de photons unique qui ajoute un photon à l'état de vide  $|0\rangle$  et fait ainsi passer tout mode de vide à l'état  $|1\rangle$ . Ce processus est non déterministe mais il est suffisant pour le calcul quantique. Les éléments optiques les plus simples sont les déphaseurs (*phase-shifters*) et les séparateurs de faisceaux (*beam splitters*). Ces éléments sont utilisés pour agir comme des opérations de porte sur nos états préparés. Comme ces deux transformations sont unitaires, nous pouvons écrire chacun de ces éléments en termes de matrice unitaire. Une matrice unitaire de déphaseur, agissant sur un seul mode avec  $N$  est l'opérateur nombre de particule, est simplement :

$$P_\phi = e^{iN\phi}$$

La matrice unitaire d'un diviseur de faisceau est donnée par

$$B_{\theta,\phi} = \begin{pmatrix} \cos\theta & -e^{i\phi}\sin\theta \\ e^{i\phi}\sin\theta & \cos\theta \end{pmatrix}$$

dans la base des modes optiques.  $\theta$  représente le biais du séparateur et  $\phi$  donne la relation de phase

## 2.3 Boson random sampling

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

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] :

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

Ainsi, on observe qu'à la sortie du circuit linéaire une densité de probabilité dépendant du circuit linéaire  $U$ .  $U$  est une matrice unitaire.

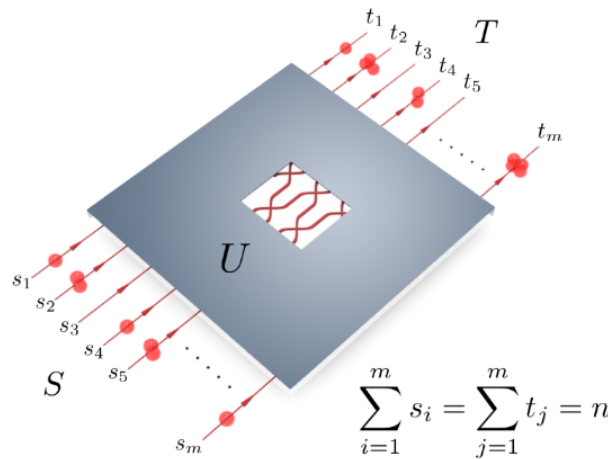


FIGURE 1 – Circuit du Boson random sampling

Il y a conservation du nombre de photons.

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

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

$\psi$  possède 2 propriétés :

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

## 2.4 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.

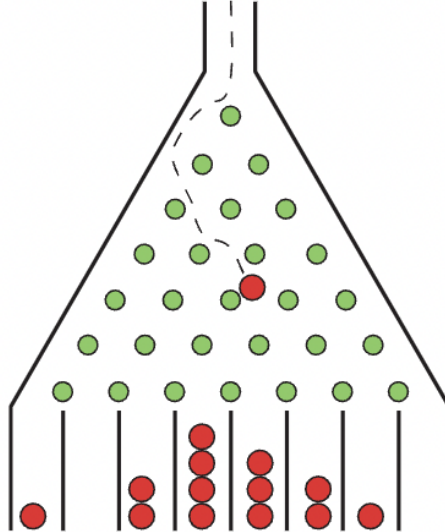


FIGURE 2 – Galton Board

## 2.5 Probabilité de sortie : apparition du permanent

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  $U$  a été démontré pour la première fois par Sheel [4]

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

avec

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

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,  $P = NP$ .

### 2.5.1 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  $m$  modes, 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 la passage de l'état dans le circuit optique. [5]

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

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

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  $V^\dagger$  de part et d'autre des coefficients multiplicatifs. Ils se s'annule 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  $|\phi\rangle = \prod_{i=1}^m \frac{(b_i^\dagger)^{t_i}}{\sqrt{t_i!}} |0\rangle$  :

$$= \langle 0 | V V^\dagger \left( \prod_{i=1}^m \frac{a_i^{t_i}}{\sqrt{t_i!}} \right) V |\psi_{in}\rangle = \langle 0 | \prod_{i=1}^m \frac{(V^\dagger a_i V)^{t_i}}{\sqrt{t_i!}} |\psi_{in}\rangle = \langle 0 | \prod_{i=1}^m \frac{b_i^{t_i}}{\sqrt{t_i!}} |\psi_{in}\rangle = \langle \phi | \psi_{in} \rangle$$

Par conséquent, d'après la définition du modèle du Boson random sampling , cela nous revient à calculer [4] :

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

### 2.5.2 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  $(2 \times 2)$  :

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

Comme décrit plus haut, nous avons besoin de calculer le vecteur  $|\phi\rangle$  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 " $|\phi\rangle$ " associé :

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

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 :

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

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

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

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

$$|\langle 11 | 11 \rangle|^2 = \left| \text{Perm} \begin{pmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{pmatrix} \right|^2$$

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  $U$ .



### 2.5.3 Cas général

Partons désormais du fait que nous avons à calculer :

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

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

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

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 :

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

Maintenant, on intervertit la somme et le produit

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

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

Et on obtient :

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

Maintenant qu'on a cela, on sait que :

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

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

$$\begin{aligned}
|\phi\rangle &= \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\prod_{i=1}^m (t_i!)^{\frac{1}{2}}}{\prod_{i,j=1}^m t_{ij}!} \left( \prod_{i,j} (U_{ji})^{t_{ij}} \right) \prod_{i=1}^m \sqrt{(\sum_j t_{ij})!} \sum_j t_{1j} \sum_j t_{2j} \cdots \sum_j t_{mj} \rangle \\
&= \sum_{\substack{\{t_{kj}\} \\ \sum_k t_{kj} = t_j}} \frac{\prod_{i=1}^m (s_i!)^{\frac{1}{2}} \prod_{i=1}^m (t_i!)^{\frac{1}{2}}}{\prod_{i,j=1}^m t_{ij}!} \left( \prod_{i,j} (U_{ji})^{t_{ij}} \right) |s_1 s_2 \dots s_m\rangle
\end{aligned}$$

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  $i$ ème colonne et  $t_j$  fois la  $j$ ème ligne de la matrice  $U$ .

Pour être plus précis, et en utilisant les notations introduise par Scheel [4],  $U_{S,T} = U[(s_1 s_2 \dots s_m), (t_1 t_2 \dots t_m)] = (U_{s_i, t_j})_{i,j \in \{1 \dots m\}}$ .

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

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

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  $m$  modes et ce seront les  $n$  premiers modes. L'état de Fock d'entrée sera  $|s_1 s_2 \dots s_m\rangle = |1_n\rangle = |11 \dots 100 \dots 0\rangle$

## 2.6 Analogie en Théorie des graphes

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

Soit  $G$  un graphe bipartie  $m \times m$ . Si 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  $U$  et  $V$  et son permanent est le nombre de match parfait. Calculer le permanent d'une matrice  $A$  une matrice avec que des 0 et des 1 est une tâche P#-complet. Ainsi, cela reste une tâche très complexe, même pour une matrice plus simple.

Le Boson random sampling et ce problème de théorie de graphe sont liés en considérant l'ensemble  $U$  qui est l'ensemble des modes d'entrées du Boson sampling et  $V$  l'ensemble des modes 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 [A&C].

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 le mode 1 et d'un photon dans le mode 2 (état  $|11\rangle$ ), il faudra que le photon dans le mode 1 reste dans le mode 1 et que le photon dans le mode 2 passe dans le mode 1.

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

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

- Soit le photon dans le mode 1 reste dans le mode 1 et le photon dans le mode 2 reste dans le mode 2.
- Soit le photon dans le mode 1 passe dans le mode 2 et le photon dans le mode 2 passe dans le mode 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 au problème de décision et au problème 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 problème 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 expert du domaine de la complexité pencherai plus pour que  $P \neq NP$  et nous prendrons cette information importante pour décrire le théorème d'Aaronson & Arkhipov

### 3.1.2 PP, BPP, BQP

Pour comparer la théorie de la complexité des algorithmes classique versus quantique, il faut ajouter dans les définitions une part de probabilité et de hasard. Ainsi, on compare des algorithmes quantiques avec des algorithmes classiques randomisé 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 problème de décision en temps polynomiale, 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  $\varepsilon$  (pour tout  $\varepsilon$  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 :

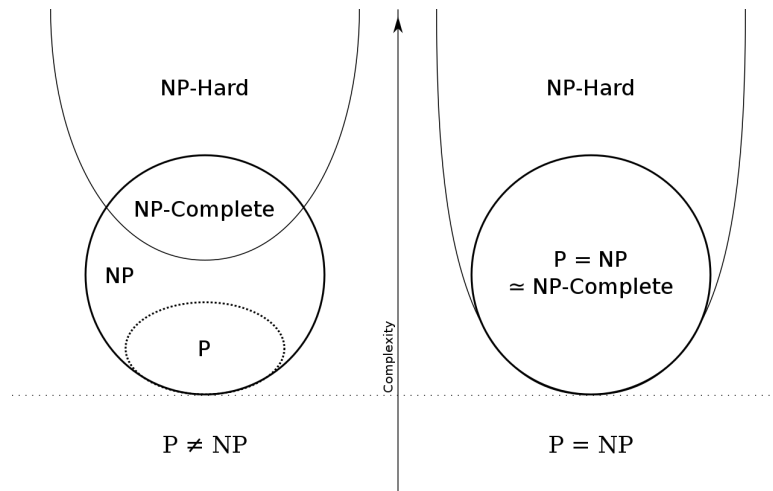


FIGURE 3 – Diagramme d'Euler

**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 (boite 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 :

$$\Sigma_0^P = P$$

$$\forall i \in \mathbb{N}^*, \Sigma_{i+1}^P = NP^{\Sigma_i^P}$$

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  $\Sigma_i$  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 :

**Théorème 1.** 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.

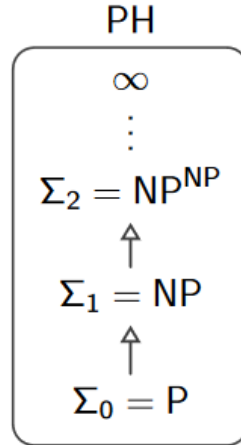


FIGURE 4 – Hiérarchie polynomiale

### 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 2** (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

Avec le Boson random sampling, nous savons que nous pouvons échantillonner avec un circuit linéaire optique  $U$  une densité de probabilité dépendant du permanent d'une certaine matrice  $U_{S,T}$ . Ainsi, comme on peut le savoir, à l'aide d'une méthode de Monte Carlo, on peut calculer le permanent de la matrice  $U$  en échantillonnant un grand nombre de fois. Un circuit linéaire optique est encodé dans une matrice unitaire  $U$ . Comme nous souhaitons plutôt faire l'inverse c'est à dire encoder une matrice  $X$  dans un circuit linéaire optique, nous pouvons nous demander comment faire. Le théorème suivant nous donne l'existence d'un tel encodage [Reck] :

**Théoreme 3.** *Soit  $U$  une matrice unitaire de taille  $(m, m)$ . On peut réaliser un circuit linéaire optique représentant cette matrice  $U$  avec  $O(m^2)$  diviseur de faisceaux et déphaseurs.*

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 diviseurs de faisceau et des déphaseurs. 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.

Supposons maintenant que nous ayons une matrice  $X \in \mathbb{C}^{nn}$ , et que nous souhaitions 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 :

**Théorème 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 1** (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 :

$$\begin{aligned} \forall z &= (z_i)_{i \in \{1, \dots, n\}} \in \{-1, 1\}^n \\ \text{Rys}_z(X) &= z_1 \dots z_n \prod_{i=1}^n (x_{i1} z_1 + \dots + x_{in} z_n) \end{aligned}$$

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 2.** Soit  $f : \llbracket 1, n \rrbracket \rightarrow \llbracket 1, n \rrbracket$  et définissons :

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

.

Alors, il n'y que 2 possibilités de résultat pour  $\eta$  :

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

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

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

□

et en utilisant la formule de Glynn :

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

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  $\varepsilon$

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

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  $\varepsilon$  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  $\varepsilon$  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 :

$$\begin{aligned}\text{Perm}(A) &= \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i, \sigma_i} \\ \det(A) &= \sum_{\sigma \in S_n} \varepsilon(\sigma) \prod_{i=1}^n a_{i, \sigma_i}\end{aligned}$$

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 :

**Théorème 7** (1971 Valiant). *Le calcul du permanent de 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.* A étudier □

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

**Théoreme 9** (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 10** (1983 Stockmayers). *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  $BPP^{NP}$ .

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. Dans ce cas, il est facile de montrer 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  $g \in \left[1 + \frac{1}{\text{poly}(n)}, \text{poly}(n)\right]$ .

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  $\varepsilon = 1/\|X\| >= 2^{\text{poly}(n)}$ ,  $\exists U \in \mathbb{K}^{2n \times 2n}$  tel que :  $U = \left( \begin{array}{c|c} Y & \\ \hline Q & \end{array} \right)$

Prenons la matrice  $A = \begin{pmatrix} Y \\ Q \end{pmatrix}$  qui correspond aux  $n$  premières colonnes de  $U$ . Alors, nous nous intéressons à la probabilité  $p_A$  d'avoir la sortie  $|1_n\rangle = |1 \dots 10 \dots 0\rangle$  pour l'entrée  $X$  dans un algorithme randomisé  $\mathcal{O}$  réalisant le Boson random sampling.

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

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 11** ((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  $\Sigma_P^3$  n'est pas censée être contenue dans  $BPP$ .  $\square$

## Références

- [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] 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).
- [5] 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).
- [6] Lance FORTNOW. « A Simple Proof of Toda's Theorem ». en. In : *THEORY OF COMPUTING* 5 (2009).