

Algorithme Q-Means

Approche Quantique pour le Clustering

Antonin LEFEVRE¹ 

¹Université de Reims

Abstract

L'algorithme K-means est une méthode de clustering très répandue en apprentissage non supervisé. Cependant, il présente des limitations, notamment une complexité temporelle en $O(iknm)$ avec i le nombre d'itérations, k le nombre de clusters, n le nombre de points et m la dimension de l'espace de définition des observations. Récemment, une variante quantique de cet algorithme, appelée Q-means [1], a été développée pour pallier ces défauts.

Cet article propose une introduction aux modèles de clustering k-means et δ -k-means ainsi qu'une introduction à l'informatique quantique et son lien étroit avec le machine learning. Est détaillé ensuite le fonctionnement de l'algorithme Q-means dans une version hybride mêlant des calculs quantiques et classiques, permettant d'être réellement implémentés. Cet article couvre le calcul de distance quantique, la recherche du minimum quantique dans une liste, et la version quantique de la méthode d'initialisation k-means++, ainsi que leurs formulations mathématiques respectives, conceptions de circuits et implémentation avec Qiskit.

Plain Language Summary

Cet article explore le fonctionnement du K-means quantique, appelé Q-Means, une méthode avancée pour regrouper des données similaires. Grâce à l'application de principes issus de la physique quantique, cette technique est capable de traiter des ensembles de données plus rapidement et avec une plus grande précision que les méthodes traditionnelles.

Keywords Q-means, Quantique, Clustering, Apprentissage Automatique, Optimisation

Key Points

- K-means et δ -k-means
- Physique quantiques
- Introduction Q-means

Correspondence to

Antonin LEFEVRE
antoninlefevre45@icloud.com

Data Availability

Tous les codes sont disponibles sur [GitHub](#).

TABLE DES MATIÈRES

| | |
|---|----|
| 1. Introduction au k-means | 3 |
| 2. Introduction au δ-k-means | 4 |
| 3. Techniques d'initialisation des clusters | 5 |
| 3.1. Maximin et ses variantes | 5 |
| 3.2. k-means++ et ses variantes | 5 |
| 4. De la physique quantique au q-means | 6 |
| 4.1. La physique quantique | 6 |
| 4.2. L'informatique quantique de base | 6 |
| 4.3. Les opérateurs quantiques | 8 |
| 4.4. Exemples de portes quantiques | 10 |
| 4.5. Projecteur, mesure et espérance d'un observable | 16 |
| 4.6. Oracle | 18 |
| 4.7. Exemple d'algorithme : l'algorithme de Grover | 19 |
| 4.8. Ordinateur quantique | 24 |
| 4.9. QRAM | 26 |
| 4.10. Le Q-Means: La Rencontre du Quantique et du Clustering | 26 |
| 5. Procédures Quantiques pour l'Algorithme Q-means | 27 |
| 5.1. Estimation quantique de la distance euclidienne entre des points | 27 |
| 5.2. Comparer deux entiers | 35 |
| 5.3. Calcul quantique du minimum d'une liste d'entiers | 41 |
| 6. Algorithme Q-Means | 51 |
| 6.1. Initialisation Q-Means++ | 51 |
| 6.2. Estimation de la distance des centroïdes | 51 |
| 6.3. Affectation du cluster | 51 |
| 7. Pseudo-code Q-Means | 52 |
| 8. Discussion | 53 |
| 8.1. Restriction sur le nombre de qubits | 53 |
| 8.2. Nombre d'itérations de l'opérateur \hat{P} | 53 |
| 8.3. Implémentation | 53 |
| 8.4. État de l'art de l'algorithme | 53 |
| 9. Conclusion | 54 |
| References | 55 |

1.. INTRODUCTION AU K-MEANS

L'algorithme **k-means** résout un problème d'optimisation en essayant de minimiser la fonction d'objectif, souvent appelée "inertie" ou "fonction coût", définie comme :

$$J(C, c_1, c_2, \dots, c_k) = \sum_{i=1}^k \sum_{x \in C_i} \|x - c_i\|^2 \quad (1)$$

où C est l'ensemble de k clusters C_1, C_2, \dots, C_k , et c_i est le centroïde du cluster C_i .

Étapes du k-means

Étape 1. **Initialisation des centroïdes** : Plusieurs méthodes d'initialisation existent. Allant de l'initialisation aléatoire des centroïdes à des techniques plus complexes comme le k-means++ [2], expliquée dans la [Section 3](#).

Étape 2. **Affectation des points aux clusters** : On utilise souvent la distance euclidienne $\|x - c_i\|$, bien que d'autres métriques comme la distance de Manhattan puissent également être utilisées. Le point x est assigné au cluster qui minimise cette distance, on cherche ainsi :

$$\arg \min_{c_i \in C} \|x - c_i\|^2 \quad (2)$$

Étape 3. **Mise à jour des centroïdes** : Chaque nouveau centroïde est le barycentre de l'ensemble de points du cluster correspondant. Formellement :

$$c_i = \frac{1}{\#C_i} \sum_{x \in C_i} x \quad (3)$$

Avec $\#C_i$ le nombre de données du cluster C_i .

Étape 4. **Convergence** : L'algorithme itère les étapes 2 et 3 jusqu'à ce que la variation des positions des centroïdes soit inférieure à un seuil ε prédéfini, ou jusqu'à un nombre maximum d'itérations.

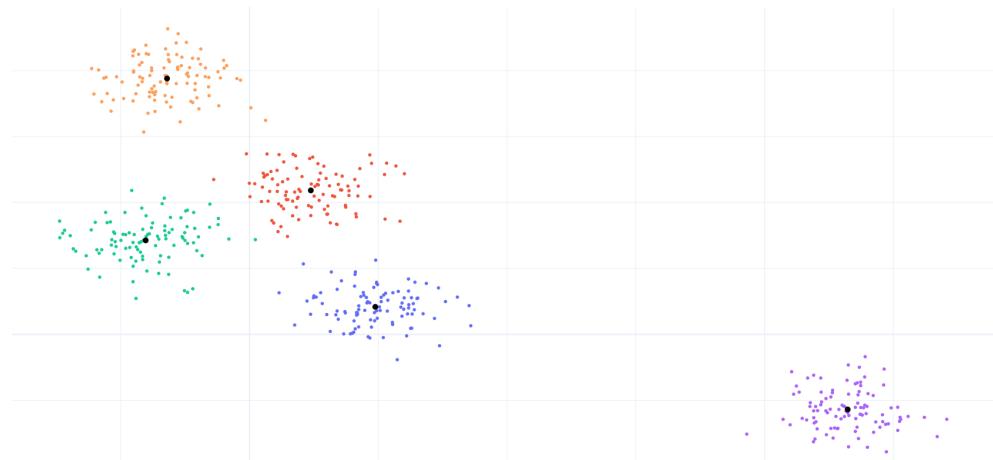


Figure 1: Exemple de clustering de données facilement séparables

2.. INTRODUCTION AU δ -K-MEANS

Le **δ -k-means** est une variante du k-means qui ajoute un paramètre de bruit δ aux étapes d'affectation et de mise à jour des centroïdes. Cette approche est particulièrement utile lorsque les données contiennent du bruit ou des valeurs aberrantes. Le **q-means** est l'analogue quantique de cet algorithme grâce au caractère non déterministe et bruité des calculs quantiques. Cela permet au Q-Means de capturer l'essence des caractéristiques de robustesse du δ -k-means tout en exploitant les avantages computationnels offerts par la physique quantique.

Ainsi, voici le détail des deux étapes qui diffèrent par rapport au **k-means**.

Assignation des étiquettes : Contrairement au k-means, l'assignation n'est pas déterministe. On crée un ensemble de labels potentiels $L_\delta(x)$:

$$L_\delta(x) = \{p : \|c_i^* - x\|^2 - \|c_p - x\|^2 \leq \delta\} \quad (4)$$

Avec c_i^* le centroïde le plus proche de la donnée x . Un label est choisi aléatoirement parmi cet ensemble.

Pour être plus précis, $L_\delta(x)$ est l'ensemble de tous les indices p des centroïdes tels que la distance au carré entre le point x et le centroïde c_p n'est pas trop différente de la distance au carré entre x et le centroïde le plus proche c_i^* . La quantité δ est la tolérance que l'on permet pour cette différence.

L'idée est que si un point x est presque équidistant de deux centroïdes ou plus, alors il pourrait raisonnablement appartenir à n'importe quel cluster associé à ces centroïdes. Le paramètre δ permet d'introduire cette flexibilité dans l'assignation des points aux clusters.

Cela rend l'algorithme plus robuste aux variations dans les données et peut être particulièrement utile dans des scénarios où les clusters ne sont pas bien séparés.

Mise à jour des centroïdes : Du bruit gaussien de variance $\frac{\delta}{2}$ est ajouté au centroïde lors de sa mise à jour :

$$c_i = \frac{1}{\#C_i} \sum_{x \in C_i} x + N\left(0, \frac{\delta}{2}\right) \quad (5)$$

Cette addition de bruit peut notamment faciliter la découverte de structures subtiles dans les données que le K-Means standard pourrait manquer.

3.. TECHNIQUES D'INITIALISATION DES CLUSTERS

La qualité des résultats de l'algorithme K-means (et de ses variantes) dépend fortement des centroïdes initiaux. Un mauvais choix peut entraîner des mauvais résultats en raison d'un optimum local. Ainsi, une initialisation aléatoire est généralement une mauvaise initialisation, mais il existe des améliorations à cette dernière, comme la méthode *Maximin* [3] ou la procédure *K-means++* [2].

3.1.. *Maximin et ses variantes*

3.1.1.. *Maximin Standard:*

La méthode Maximin est une façon simple et directe d'obtenir des centres initiaux bien dispersés. Le premier centre est choisi aléatoirement parmi les points de données. Ensuite, tous les centres suivants sont choisis de manière à maximiser la distance minimale par rapport aux centres de clusters déjà choisis. Une implémentation naïve nécessite un grand nombre de calculs de distance, mais des optimisations sont possibles pour réduire ce nombre.

3.1.2.. *Trimmed Maximin:*

Un problème majeur de la méthode Maximin est qu'elle a tendance à sélectionner des valeurs aberrantes. Une solution simple à ce problème est le "Trimmed Maximin", où un certain nombre de points les plus éloignés sont éliminés à chaque étape de sélection. Le point ayant la $(s + 1)$ -ième plus grande distance minimale aux centres déjà choisis est alors sélectionné.

3.2.. *k-means++ et ses variantes*

3.2.1.. *k-means++ Standard:*

La procédure k-means++ peut être considérée comme une version aléatoire de la méthode Maximin. Au lieu de choisir systématiquement le point ayant la plus grande distance minimale par rapport aux centres déjà choisis, ce point se voit attribuer une probabilité plus élevée d'être choisi. Plus précisément, la probabilité qu'un point soit choisi comme prochain centre est proportionnelle au carré de la distance minimale qu'il a par rapport aux centres de clusters déjà choisis. Comme pour Maximin, une implémentation standard de k-means++ nécessite un grand nombre de calculs de distance. Pour les grands ensembles de données, des méthodes d'approximation basées sur la chaîne de Markov Monte Carlo peuvent être utiles.

3.2.2.. *Trimmed k-means++:*

k-means++ présente deux inconvénients majeurs. D'abord, les valeurs aberrantes ont toujours une forte probabilité d'être choisies comme centres initiaux. Ensuite, même si les points ayant une petite distance minimale aux centres déjà choisis ont une faible probabilité d'être choisis, leur nombre augmente naturellement à mesure que de nouveaux centres sont sélectionnés. Cela peut entraîner un risque élevé de choisir des centres trop proches les uns des autres. Pour résoudre ces problèmes, une approche de "trimming" basée sur des quantiles est proposée. Les points de données ayant une distance minimale en dessous d'un quantile inférieur spécifié par l'utilisateur ou au-dessus d'un quantile supérieur ne peuvent pas être choisis comme prochains centres de cluster.

4.. DE LA PHYSIQUE QUANTIQUE AU Q-MEANS

Avant de plonger dans les subtilités du Q-means, il est crucial de saisir les principes fondamentaux de la physique quantique et de l'informatique quantique sur lesquels il s'appuie.

4.1.. La physique quantique

La physique quantique est une branche de la physique qui étudie des phénomènes à l'échelle atomique et subatomique. Contrairement à la physique classique, qui opère au niveau macroscopique, la physique quantique concerne le monde des particules élémentaires pour étudier leurs comportements souvent contre-intuitifs. Parmi les concepts clés, la superposition [4] et l'intrication [5] quantique sont fondamentales. La superposition permet à une particule d'exister simultanément dans plusieurs états, rendant son comportement beaucoup plus complexe. L'intrication quantique, quant à elle, crée des liens entre les états quantiques de deux particules ou plus, indépendamment de la distance les séparant.

4.2.. L'informatique quantique de base

L'informatique quantique est un domaine en plein essor qui utilise les principes de la physique quantique pour effectuer des calculs. Dans cette discipline, on ne s'appuie pas sur des bits traditionnels qui peuvent seulement être dans un état 0 ou 1. À la place, on utilise des qubits (bits quantiques). La particularité d'un qubit réside dans sa capacité à exister dans une superposition d'états.

Pour représenter un qubit, nous utilisons la notation Dirac. Ainsi, $|0\rangle$ est utilisé pour représenter un qubit à l'état 0 et $|1\rangle$ pour représenter un qubit dans l'état 1. Ces états peuvent être exprimés vectoriellement de la manière suivante:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (6)$$

Contrairement à un bit classique qui ne se trouve que dans un seul état à la fois, l'état d'un qubit, noté $|\psi\rangle$, peut être à la fois dans l'état $|0\rangle$ et en même temps dans l'état $|1\rangle$ avec différentes probabilités. L'état d'un qubit est exprimé comme ceci:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (7)$$

Où α et β sont des nombres complexes qui doivent satisfaire la condition $|\alpha|^2 + |\beta|^2 = 1$. En fait, la mesure de $|\psi\rangle$ renvoie soit $|0\rangle$ avec une probabilité $|\alpha|^2$ soit $|1\rangle$ avec une probabilité $|\beta|^2$. On peut également décrire l'état d'un qubit sous forme d'équation:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (8)$$

Géométriquement, on représente un qubit dans la sphère de Bloch, illustrée dans la [Figure 2](#). Pour se faire, on exprime les coefficients α et β en coordonnées polaires.

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\varphi}|1\rangle \quad (9)$$

C'est la représentation d'un qubit avec un modèle à deux paramètres où $\theta \in [0, \pi]$ et $\varphi \in [0, 2\pi]$.

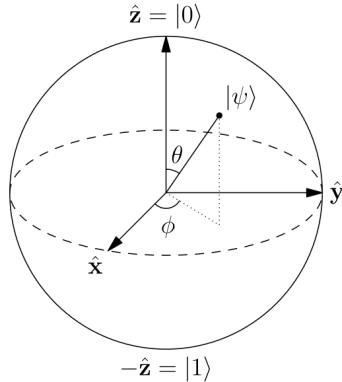


Figure 2: Sphère de Bloch.

On peut aussi avoir des systèmes multi-qubits, qui sont représentés comme le produit tensoriel de qubits uniques. Par exemple l'état d'un système à deux qubits $|\psi_1\rangle = \lambda_0|0\rangle + \lambda_1|1\rangle$ et $|\psi_2\rangle = \lambda_2|0\rangle + \lambda_3|1\rangle$ s'exprime comme:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = (\lambda_0|0\rangle + \lambda_1|1\rangle) \otimes (\lambda_2|0\rangle + \lambda_3|1\rangle) = \begin{pmatrix} \lambda_0\lambda_2 = \alpha_0 \\ \lambda_0\lambda_3 = \alpha_1 \\ \lambda_1\lambda_2 = \alpha_2 \\ \lambda_1\lambda_3 = \alpha_3 \end{pmatrix} \quad (10)$$

Qui se note sous forme d'équation de la manière suivante:

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \quad (11)$$

Où $|\alpha_0|^2$ est la probabilité de mesurer les deux qubits dans l'état $|0\rangle$, $|\alpha_1|^2$ est la probabilité de mesurer le premier qubit dans l'état $|0\rangle$ et le second dans l'état $|1\rangle$, et ainsi de suite. Comme précédemment, nous avons:

$$|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 + |\alpha_3|^2 = 1 \quad (12)$$

De manière générale, un système à N qubits peut s'écrire comme la superposition de N états $|i\rangle$:

$$|\psi\rangle = \sum_{i=0}^{N-1} \alpha_i|i\rangle \quad (13)$$

4.3.. Les opérateurs quantiques

En informatique quantique, la connexion entre les principes de la physique quantique et leur application pratique est souvent représentée par les circuits quantiques. Comme en informatique classique, où nous utilisons des circuits logiques pour effectuer des calculs, les circuits quantiques servent de plateforme pour manipuler des états quantiques. Ils sont constitués d'une séquence d'opérations quantiques, appelées portes quantiques, qui agissent sur un ensemble de qubits.

Une porte quantique est un opérateur linéaire noté \hat{U} qui permet de passer d'un état $|\Phi\rangle$ à un état $|\Psi\rangle$ tel que :

$$|\Psi\rangle = \hat{U}|\Phi\rangle \quad (14)$$

Pour un système à n qubits, $|\Phi\rangle$ et $|\Psi\rangle$ sont des vecteurs de taille 2^n et l'opérateur \hat{U} est une matrice de $\mathcal{M}_{2^n}(\mathbb{C})$ ayant certaines propriétés, comme par exemple :

- **Linéarité** : Les opérateurs unitaires sont linéaires. Cela signifie que pour tout état quantique $|\psi_1\rangle$ et $|\psi_2\rangle$ et pour tous les scalaires complexes c_1 et c_2 , l'opération suivante est vraie :

$$\hat{U}(c_1|\psi_1\rangle + c_2|\psi_2\rangle) = c_1\hat{U}|\psi_1\rangle + c_2\hat{U}|\psi_2\rangle \quad (15)$$

Cette propriété est fondamentale pour les opérations en mécanique quantique et garantit que l'action de l'opérateur est distributive et homogène.

- **Unitarité** : La définition même d'un opérateur unitaire est basée sur la propriété d'unitarité. Un opérateur \hat{U} est unitaire si, et seulement si, son adjoint (ou conjugué-transposé, noté \hat{U}^\dagger) est son inverse. Mathématiquement, cela se traduit par:

$$\hat{U}^\dagger\hat{U} = \hat{U}\hat{U}^\dagger = \hat{I} \quad (16)$$

où \hat{I} est l'opérateur identité.

- **Inversabilité** : Les opérateurs unitaires sont inversibles, et leur inverse est donné par l'adjoint (ou le conjugué-transposé) de l'opérateur. C'est-à-dire que :

$$\hat{U}^{-1} = \hat{U}^\dagger \quad (17)$$

- **Produit de deux opérateurs unitaires** : Si \hat{U} et \hat{V} sont deux opérateurs unitaires, alors leur produit $\hat{U}\hat{V}$ est également unitaire.
- **Tenseur des opérateurs unitaires** : Si \hat{U} et \hat{V} sont unitaires, alors leur produit tensoriel $\hat{U} \otimes \hat{V}$ est également unitaire.
- **Valeurs propres** : Les valeurs propres (ou eigenvalues) d'un opérateur unitaire ont une magnitude de 1. C'est-à-dire que si λ est une valeur propre de \hat{U} , alors $|\lambda| = 1$.

En utilisant plusieurs opérateurs, on peut construire des circuits quantiques. Par exemple:

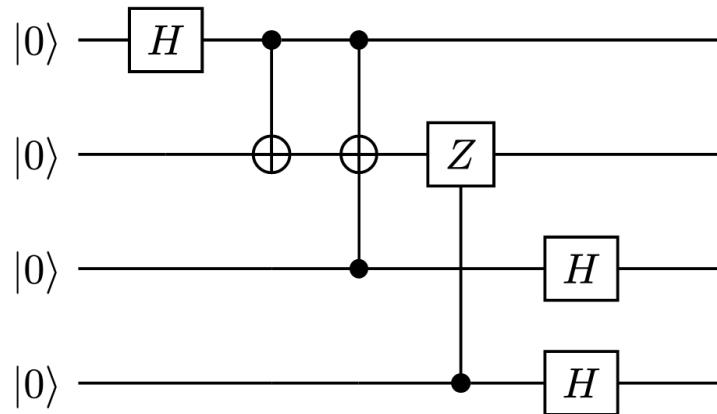


Figure 3: Exemple de circuit quantique à 4 qubits.

Chaque ligne horizontale du circuit représente un qubit. Les portes quantiques, sont appliquées aux qubits de gauche à droite sur le circuit. Les portes sont représentées par des symboles spécifiques tels que des rectangles, des cercles ou des croix. Lorsque plusieurs qubits sont impliqués dans une opération, comme c'est le cas pour les portes contrôlées, un point noir (ou plusieurs) sur une ligne indique quel qubit sert de qubit de contrôle. L'opération est appliquée au(x) qubit(s) cible(s) seulement si le(s) qubit(s) de contrôle est ou sont dans un état particulier (généralement l'état $|1\rangle$).

Attention, si un qubit étant initialement dans l'état $|\psi\rangle$ subit deux opérateurs \hat{V} puis \hat{U} à la suite dans le circuit quantique, alors l'état final est $\hat{U}\hat{V}|\psi\rangle$. En effet, la convention d'écriture pour les opérateurs linéaires est d'écrire l'opérateur à gauche de l'état sur lequel il agit. C'est pour cela que l'ordre des symboles $|\psi\rangle$, \hat{V} et \hat{U} en partant de la gauche du circuit est inversé par rapport à l'ordre dans lequel ils apparaissent dans l'écriture algébrique de l'état final.

4.4.. Exemples de portes quantiques

4.4.1.. Porte de Hadamard (porte **H**):

La porte de Hadamard est une des portes quantiques les plus fondamentales et est souvent utilisée pour créer une superposition de qubits. Elle est représentée par la matrice suivante:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (18)$$

Son symbole est:



Application sur $|0\rangle$:

Si vous appliquez la porte de Hadamard à un qubit dans l'état $|0\rangle$, vous obtiendrez une superposition des états $|0\rangle$ et $|1\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (19)$$

Application sur $|1\rangle$:

De même, si vous appliquez la porte de Hadamard à un qubit dans l'état $|1\rangle$, vous obtiendrez :

$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (20)$$

Application sur un état en superposition:

Imaginons que nous avons un qubit dans une superposition d'états $\alpha|0\rangle + \beta|1\rangle$. Appliquer une porte de Hadamard à cet état donnera :

$$H(\alpha|0\rangle + \beta|1\rangle) = \frac{\alpha + \beta}{\sqrt{2}}|0\rangle + \frac{\alpha - \beta}{\sqrt{2}}|1\rangle \quad (21)$$

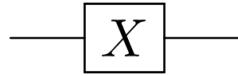
Ces propriétés rendent la porte de Hadamard extrêmement utile pour créer des états quantiques en superposition, ce qui est souvent une première étape nécessaire dans de nombreux algorithmes quantiques.

4.4.2.. Porte de **Pauli-X** (Porte **NOT**):

La porte X est souvent appelée la porte **NOT** quantique car elle agit sur un qubit en inversant son état. Elle est représentée par la matrice suivante :

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (22)$$

Son symbole est:



Application sur $|0\rangle$:

Si nous appliquons la porte X à un qubit dans l'état $|0\rangle$, alors:

$$X|0\rangle = |1\rangle \quad (23)$$

Application sur $|1\rangle$:

De la même manière, si nous appliquons la porte X à un qubit dans l'état $|1\rangle$, alors:

$$X|1\rangle = |0\rangle \quad (24)$$

Application sur état en superposition:

Supposons que nous avons un qubit dans une superposition d'états $\alpha|0\rangle + \beta|1\rangle$. Appliquer une porte X à cet état donnera:

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle \quad (25)$$

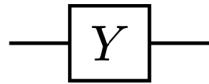
La porte X est souvent utilisée pour réaliser des inversions dans les algorithmes quantiques. Elle est particulièrement utile pour manipuler et contrôler l'état de qubits individuels.

4.4.3.. Porte de **Pauli-Y**:

La porte Y est une autre porte quantique de Pauli qui effectue une rotation sur le qubit tout en changeant les phases. Elle est représentée par la matrice suivante :

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad (26)$$

Son symbole est:



Application sur $|0\rangle$:

Si nous appliquons la porte Y à un qubit dans l'état $|0\rangle$, alors:

$$Y|0\rangle = -i|1\rangle \quad (27)$$

Application sur $|1\rangle$:

De la même manière, si nous appliquons la porte Y à un qubit dans l'état $|1\rangle$, alors:

$$Y|1\rangle = i|0\rangle \quad (28)$$

Application sur un état en superposition:

Supposons que nous avons un qubit dans une superposition d'états $\alpha|0\rangle + \beta|1\rangle$. Appliquer une porte Y à cet état donnera:

$$Y(\alpha|0\rangle + \beta|1\rangle) = -i\alpha|1\rangle + i\beta|0\rangle \quad (29)$$

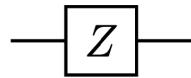
La porte Y est utile pour des opérations qui nécessitent des changements de phase et des rotations. Elle joue un rôle dans divers algorithmes et circuits quantiques.

4.4.4.. Porte de Pauli-Z:

La porte Z est une autre porte quantique fondamentale, aussi appelée la porte de Pauli-Z. Elle effectue une rotation de π radians autour de l'axe Z de la sphère de Bloch. Sa matrice représentative est :

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (30)$$

Son symbole est:



Application sur $|0\rangle$:

Si nous appliquons la porte Z à un qubit dans l'état $|0\rangle$, alors:

$$Z|0\rangle = |0\rangle \quad (31)$$

Application sur $|1\rangle$:

De la même manière, si nous appliquons la porte Z à un qubit dans l'état $|1\rangle$, alors:

$$Z|1\rangle = -|1\rangle \quad (32)$$

Application sur un état en superposition:

Supposons que nous avons un qubit dans une superposition d'états $\alpha|0\rangle + \beta|1\rangle$. Appliquer une porte Z à cet état donnera:

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle - \beta|0\rangle \quad (33)$$

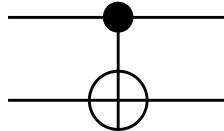
La porte Z est utilisée dans plusieurs algorithmes quantiques pour effectuer des changements de phases. Elle est également utile pour des opérations comme la création de qubits de contrôle dans des circuits plus complexes.

4.4.5.. Porte CNOT:

La porte CNOT, ou “Controlled NOT”, est une porte quantique à deux qubits qui effectue une opération de NOT (aussi appelée opération X) sur un qubit cible (symbole de croix) si un qubit de contrôle (le point noir) est dans l'état $|1\rangle$. La matrice représentative de la porte CNOT est la suivante :

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (34)$$

Son symbole est:



Qubit de contrôle à $|0\rangle$:

Dans le cas où le qubit de contrôle (le premier) est dans l'état $|0\rangle$ alors le qubit cible ne change pas. Ainsi:

$$\text{CNOT}|00\rangle = |00\rangle \quad \text{CNOT}|01\rangle = |01\rangle \quad (35)$$

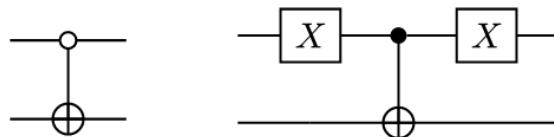
Qubit de contrôle à $|1\rangle$:

Dans le cas où le qubit de contrôle (le premier) est dans l'état $|1\rangle$ alors on applique la porte NOT au qubit cible. Ainsi:

$$\text{CNOT}|10\rangle = |11\rangle \quad \text{CNOT}|11\rangle = |10\rangle \quad (36)$$

La porte CNOT est fondamentale dans la construction de circuits quantiques complexes et est souvent utilisée pour créer de l'intrication entre les qubits.

À noter l'équivalence suivante:



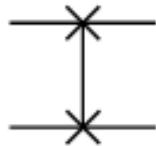
Qui permet de déclencher la porte NOT si le qubit de contrôle est à $|0\rangle$.

4.4.6.. Porte **SWAP**:

La porte SWAP est une porte quantique à deux qubits qui échange les états des qubits sur lesquels elle est appliquée. On peut ajouter des bits de contrôle si nécessaire. Si on applique la porte SWAP à deux qubits, l'état du premier qubit devient l'état du second qubit, et vice versa. La matrice représentative de la porte SWAP est la suivante :

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (37)$$

Son symbole est:



Échange des qubits:

Lorsqu'on applique la porte SWAP, les états des qubits sont échangés :

$$\text{SWAP}|01\rangle = |10\rangle \quad \text{SWAP}|10\rangle = |01\rangle \quad (38)$$

Qubits dans le même état:

Si les deux qubits sont dans le même état, l'application de la porte SWAP ne change pas leur état :

$$\text{SWAP}|00\rangle = |00\rangle \quad \text{SWAP}|11\rangle = |11\rangle \quad (39)$$

La porte SWAP est utile dans les circuits quantiques pour réorganiser les états des qubits sans modifier leur superposition ou leur intrication.

4.5.. Projecteur, mesure et espérance d'un observable

4.5.1.. Projecteur:

En mathématiques, et plus spécifiquement en mécanique quantique, un projecteur est un type particulier d'opérateur linéaire. En plus de respecter les propriétés classiques énoncées dans la [Section 4.3](#), on peut ajouter:

- **Idempotence:** Si P est un projecteur alors $P^2 = P$.
- **Orthogonalité:** Si P est un projecteur orthogonal dans un espace vectoriel \mathbb{V} , alors $\forall v \in \mathbb{V}, P(v)$ est orthogonal au vecteur $v - P(v)$.

À noter qu'un projecteur \hat{P} sur l'état $|\psi\rangle$ est $\hat{P} = |\psi\rangle\langle\psi|$.

On peut ensuite définir un ensemble complet d'opérateurs de projection dans un espace de Hilbert \mathbb{H} de dimension n , noté $\{\hat{P}_i\}$. Dans cet ensemble, les projecteurs sont deux à deux orthogonaux, c'est à dire que soit P_i et P_j deux projecteurs, $P_i P_j = 0$ si $i \neq j$. De plus, la somme de tous les projecteurs dans l'ensemble \mathbb{H} est l'opérateur identité, c'est à dire $\sum_i \hat{P}_i = \hat{I}$.

Par exemple dans \mathbb{C}^2 , les opérateurs de projection sont \hat{P}_0 et \hat{P}_1 tel que:

$$\hat{P}_0 = |0\rangle\langle 0| \quad \text{et} \quad \hat{P}_1 = |1\rangle\langle 1| \quad (40)$$

4.5.2.. Mesure d'un observable:

Lorsque qu'il y a une mesure sur un état $|\psi\rangle$, on dit qu'il y a réduction du paquet d'onde. Pour calculer mathématiquement la probabilité de mesurer un certain état i on utilise un opérateur P_i observable, pour calculer:

$$P_{R[i]} = \langle\psi|\hat{P}_i|\psi\rangle \quad (41)$$

Au final, l'état observé $|\psi'\rangle$ est:

$$|\psi'\rangle = \frac{\hat{P}_i|\psi\rangle}{\sqrt{\langle\psi|\hat{P}_i|\psi\rangle}} \quad (42)$$

Qui est bien un état normalisé.

Exemple

Soit $|\psi\rangle = \frac{\sqrt{2+i}}{\sqrt{20}}|000\rangle + \frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{10}}|011\rangle + \frac{i}{2}|111\rangle$ un système à 3 qubits.

On peut facilement connaître la probabilité que le système soit dans l'état $|011\rangle$ en calculant $|\langle 011|\psi\rangle|^2$ qui revient à calculer le module au carré du coefficient correspondant à $|011\rangle$. Ainsi,

$$\begin{aligned} P_{r[011]} &= |\langle 011|\psi\rangle|^2 \\ &= \left| \frac{1}{\sqrt{10}} \right|^2 \\ &= \frac{1}{10} \end{aligned} \quad (43)$$

Donc la probabilité de trouver le système dans l'état $|011\rangle$ est de 10%.

Si on veut maintenant calculer la probabilité que le premier qubit soit $|0\rangle$ on utilise la formule précédente $P_{R[i]} = \langle\psi|\hat{P}_i|\psi\rangle$. Comme notre système compte 3 qubits, notre opérateur \hat{P}_i est $\hat{P}_0 \otimes \hat{I} \otimes \hat{I}$. C'est à dire que le premier qubit est $|0\rangle$ et les autres ne sont pas mesurés. Ainsi,

$$\begin{aligned}
 P_{r[0]} &= \langle\psi|\hat{P}_0 \otimes \hat{I} \otimes \hat{I}|\psi\rangle \\
 &= \left(\frac{\sqrt{2}-i}{\sqrt{20}}\langle000| + \frac{1}{\sqrt{2}}\langle001| + \frac{1}{\sqrt{10}}\langle011| - \frac{i}{2}\langle111| \right) \left(\frac{\sqrt{2}+i}{\sqrt{20}}|000\rangle + \frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{10}}|011\rangle \right) \\
 &= \frac{(\sqrt{2}-i)(\sqrt{2}+i)}{20} + \frac{1}{2} + \frac{1}{10} \\
 &= \frac{3}{4}
 \end{aligned} \tag{44}$$

Donc, la probabilité que le premier qubit soit mesuré dans l'état $|0\rangle$ est de 75%. On peut maintenant exprimer l'état du système après cette mesure, qui est:

$$\begin{aligned}
 |\psi'\rangle &= \frac{\hat{P}_0 \otimes \hat{I} \otimes \hat{I}|\psi\rangle}{\sqrt{\langle\psi|\hat{P}_0 \otimes \hat{I} \otimes \hat{I}|\psi\rangle}} \\
 &= \frac{\frac{\sqrt{2}+i}{\sqrt{20}}|000\rangle + \frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{10}}|011\rangle}{\sqrt{\frac{3}{4}}} \\
 &= \frac{\sqrt{8} + \sqrt{4} * i}{\sqrt{60}}|000\rangle + \frac{\sqrt{4}}{\sqrt{6}}|001\rangle + \frac{\sqrt{4}}{\sqrt{30}}|011\rangle
 \end{aligned} \tag{45}$$

Voilà donc l'état du système après avoir mesuré le premier qubit à $|0\rangle$. Ce nouvel état est normalisé.

4.5.3.. **Espérance d'un observable:**

Lorsqu'une série de mesures d'un observable A est effectuée sur un système quantique $|\psi\rangle$, alors le résultat espéré de cette suite de mesures est l'espérance mathématique (notée $\langle\hat{A}\rangle$).

Elle s'écrit:

$$\langle\hat{A}\rangle = \langle\psi|\hat{A}|\psi\rangle \tag{46}$$

4.6.. Oracle

Un autre concept clé dans la création d'algorithmes quantiques est ce qu'on appelle les oracles. Un oracle est une boîte noire qui évalue une fonction booléenne f pour une superposition d'entrées. Formellement, l'oracle est représenté comme une fonction de \mathbb{B} , l'ensemble des vecteurs de l'espace de Hilbert, vers $\{0, 1\}$:

$$f : \mathbb{B} \rightarrow \{0; 1\} \quad (47)$$

Dans un circuit, l'oracle est modélisé par une porte unitaire U_f qui réalise l'opération suivante sur un état quantique de la forme $|x\rangle|y\rangle$:

$$U_f|x_1 \dots x_{n-1}, y\rangle = |x_1 \dots x_{n-1}, y \oplus f(x_1 \dots x_{n-1})\rangle \quad (48)$$

où \oplus représente le XOR, et $|x\rangle$ et $|y\rangle$ sont respectivement les états des qubits d'entrée et auxiliaires.

Pour exploiter l'avantage quantique, les qubits d'entrée sont d'abord préparés dans une superposition uniforme de tous les états possibles via des portes de Hadamard, transformant $|0\rangle^{\otimes n}$ en:

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \quad (49)$$

L'application de l'oracle à cet état superposé permet d'évaluer $f(x)$ pour toutes les entrées en parallèle, exploitant le parallélisme quantique.

Quantum Phase Kickback

Le *quantum phase kickback* [6] est un effet où la phase d'un état quantique est indirectement modifiée via un oracle. Si $|x\rangle$ est un état en superposition et $|y\rangle$ est un qubit auxiliaire préparé dans l'état $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, l'oracle applique une phase à $|x\rangle$ si $f(x) = 1$:

$$U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle \quad (50)$$

Cette modification de phase sur les états pour lesquels $f(x) = 1$ est exploitée par des algorithmes comme celui de Grover pour amplifier les amplitudes des états "corrects" par rapport aux autres.

Utilisation d'un oracle dans un circuit

Dans la [Figure 11](#) ci-dessous, nous observons l'utilisation d'un oracle quantique au sein d'un circuit. Les qubits d'entrée x , représentant un espace de Hilbert de dimension n , sont préparés dans un état de superposition par l'application de portes de Hadamard. L'oracle, représenté par la porte unitaire U_f , est appliqué à ces qubits d'entrée ainsi qu'à m qubits auxiliaires y , qui sont généralement initialisés à $|0\rangle$.

L'action de l'oracle est telle qu'elle réalise une transformation sur les qubits auxiliaires, conditionnée par l'état des qubits d'entrée, sans modifier l'état de x lui-même mais en ajustant sa phase si la fonction $f(x)$ évaluée est vraie. Ce phénomène est connu sous le nom de 'phase kickback'. Dans cette configuration, le symbole \oplus indique une opération de OU exclusif (XOR), qui ne modifie l'état de y que si $f(x)$ est égale à 1.

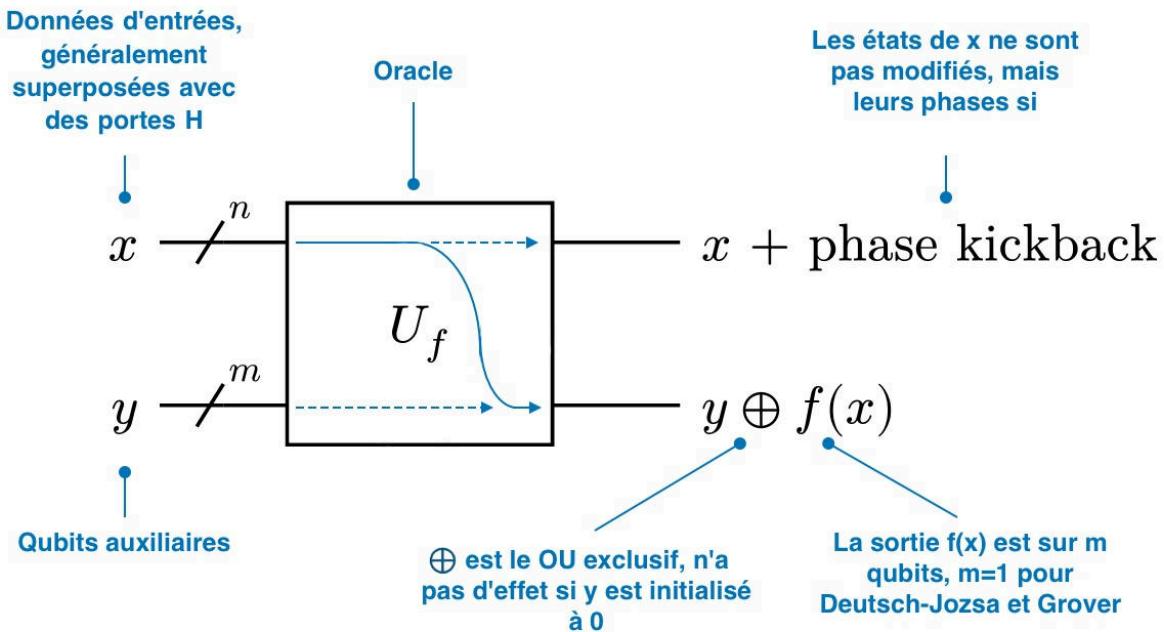


Figure 11: Utilisation d'un oracle

Il est important de noter que la sortie de la fonction $f(x)$ est représentée sur m qubits, et pour des algorithmes tels que Deutsch-Jozsa et Grover, un seul qubit auxiliaire est utilisé (c'est-à-dire $m = 1$). La mise en œuvre d'un oracle permet d'exploiter le parallélisme quantique, où la fonction est évaluée simultanément pour toutes les superpositions d'entrées possibles, ce qui est une pierre angulaire de la capacité des algorithmes quantiques à surpasser leurs homologues classiques dans certaines tâches de calcul.

4.7.. Exemple d'algorithme : l'algorithme de Grover

L'algorithme de Grover [7]-[9] est un algorithme quantique qui permet de rechercher un élément dans une liste sans avoir à parcourir tous les éléments de celle-ci. Pour ce faire, l'algorithme met à jour itérativement la loi de probabilité de sorte que, après un certain nombre d'itérations, la position de l'élément recherché soit facilement identifiable grâce à une probabilité significativement plus grande que les autres. Pour modifier cette probabilité, l'algorithme utilise un **Oracle** dont le rôle est d'identifier l'élément recherché.

4.7.1.. *Intuition:*

Nous détaillons l'intuition derrière l'algorithme de Grover dans la [Figure 12](#). Pour cet exemple nous prendrons un système à 3 qubits, ce qui représente 8 états de base, d'où les 8 rectangles.

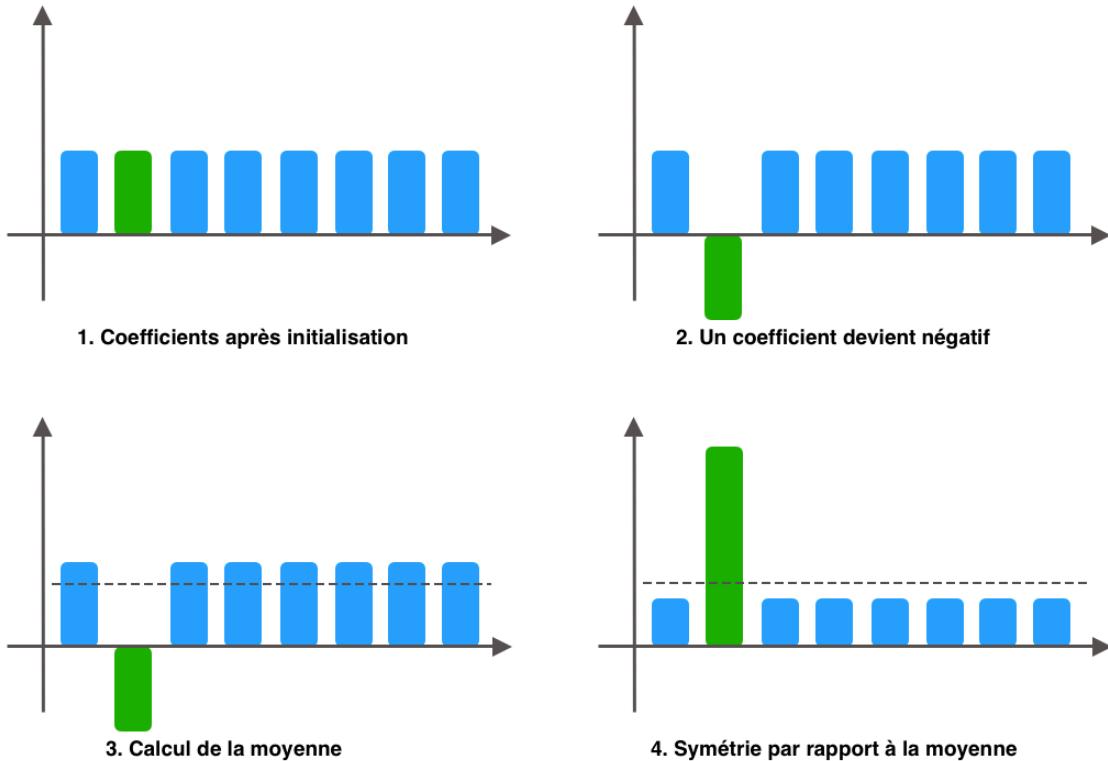


Figure 12: Les grandes étapes de l'algorithme de Grover.

- Figure 1 : Imaginons que les hauteurs des rectangles représentent les coefficients des 8 états de base. Tous les rectangles ont initialement la même hauteur, ce qui signifie que mesurer ce qubit ne favorise aucun état particulier; chaque état a une probabilité égale d'être mesuré.
- Figure 2 : Le coefficient correspondant à l'état que nous cherchons (représenté par le rectangle vert) est rendu négatif. Cela peut être fait même sans savoir quel est l'état cible. Une mesure du qubit ne serait toujours pas utile pour identifier l'état cible, car les valeurs absolues des coefficients restent égales.
- Figure 3 : La moyenne des coefficients est calculée.
- Figure 4 : Une symétrie est effectuée par rapport à cette moyenne calculée. Suite à cette opération, les rectangles bleus (états non-cibles) ont des hauteurs plus faibles, tandis que le rectangle vert (état cible) a une hauteur beaucoup plus grande. En mesurant ce nouveau qubit, il est désormais plus probable d'obtenir l'état correspondant au rectangle vert, c'est-à-dire l'état que l'on cherche.

L'algorithme de Grover consiste à répéter ce processus (les trois dernières étapes) en partant de l'état obtenu lors de la dernière itération. Après plusieurs cycles, la probabilité de mesurer l'état cible (rectangle vert) devient de plus en plus grande, ce qui facilite sa détection.

4.7.2.. **Modélisation mathématique du problème:**

La recherche dans une liste non ordonnée est modélisée à l'aide d'une fonction mathématique. Soit N un entier fixé et soit k_0 un entier avec $0 \leq k_0 \leq N - 1$. On définit alors la fonction $f : \{0, 1, \dots, N - 1\} \rightarrow \{0, 1\}$ par

$$f(k_0) = 1 \quad \text{et} \quad f(k) = 0 \quad \forall k \neq k_0 \quad (51)$$

Ainsi, l'objectif est de trouver la valeur k_0 telle que $f(k_0) = 1$. En d'autres termes on cherche l'antécédent de 1 par f . La valeur k_0 représente donc l'indice de l'élément recherché dans la liste.

4.7.3.. **Algorithm:**

Initialisation:

On initialise un registre quantique avec n qubits dans l'état $|0\rangle$ sur lesquels on applique une porte H et un qubit $n + 1$ dans l'état $|1\rangle$ sur lequel on applique également une porte H . Ainsi, l'état du système après l'initialisation est la suivante.

$$(H|0\rangle)^{\otimes n} \otimes H|1\rangle \quad (52)$$

Calculs:

La suite du circuit est une succession de ce qu'on appelle une *transformation de Grover* notée G . Cette transformation est composée d'un **Oracle** O_f et d'un opérateur de diffusion noté S_{ψ_H} . Cette porte G est itérée l fois avec $l \simeq \frac{\pi}{4}\sqrt{N}$ où $N = 2^n$. En réalité, l est exactement égale à $\frac{\pi}{4}\sqrt{N} - \frac{1}{2}$.

L'**Oracle** O_f est associé à la fonction f définie précédemment, et prend en entrée $N + 1$ qubits. L'oracle réalise une fonction $F(x, y) = (x, y \otimes f(x))$, c'est à dire que le résultat est stocké dans le $N + 1$ -ème qubit. On peut schématiser l'Oracle comme ceci:

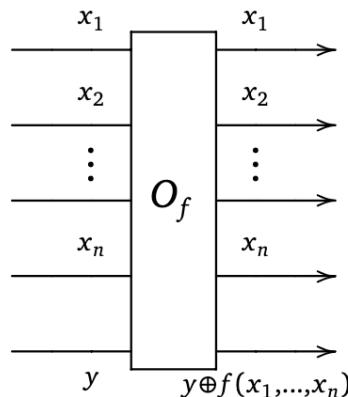


Figure 13: Oracle de l'algorithme Grover.

Cet Oracle transforme donc l'état de la dernière ligne de $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ à l'état suivant:

$$s = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes f(k) = (-1)^{f(k)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{si } k \neq k_0 \\ -\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & \text{si } k = k_0 \end{cases} \quad (53)$$

Ainsi l'oracle permet de détecter si l'entier en entrée est k_0 ou non.

L'opérateur de diffusion S_{ψ_H} lui, est appliqué après l'Oracle sur les n premiers qubits, c'est à dire ceux qui étaient initialisés à $|0\rangle$ au tout début du circuit. Il est définie par:

$$S_{\psi_H} = 2|\psi_H\rangle\langle\psi_H| - \hat{I} \quad (54)$$

On peut plus précisément décomposer la porte S_{ψ_H} en une succession de portes classiques H et Z .

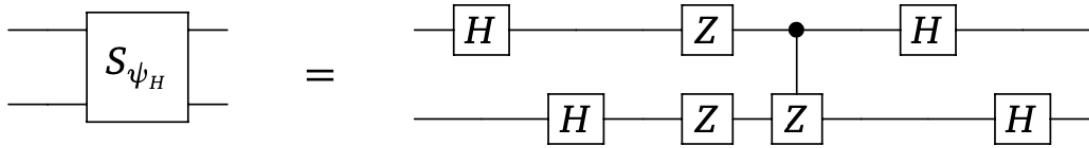


Figure 14: Décomposition de la porte S_{ψ_H} .

Finalement, on peut représenter la porte G comme la succession de l'oracle O_f et de la porte S_{ψ_H} de diffusion.

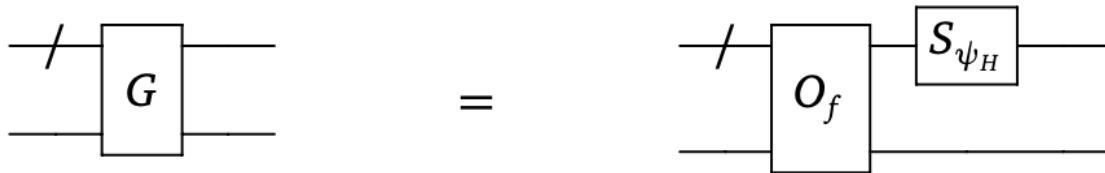


Figure 15: Décomposition de la porte de Grover.

4.7.4.. Circuit complet:

Au final, on itère l fois la porte de Grover G , et on mesure $|\psi_k\rangle$ c'est à dire les n premiers qubits du circuit. Cette mesure finale est donc la mesure d'un n -qubit (et correspond donc à n mesures de 1-qubits). Le circuit renvoie donc un n -bit classique et cet entier est très probablement le rang k_0 cherché.

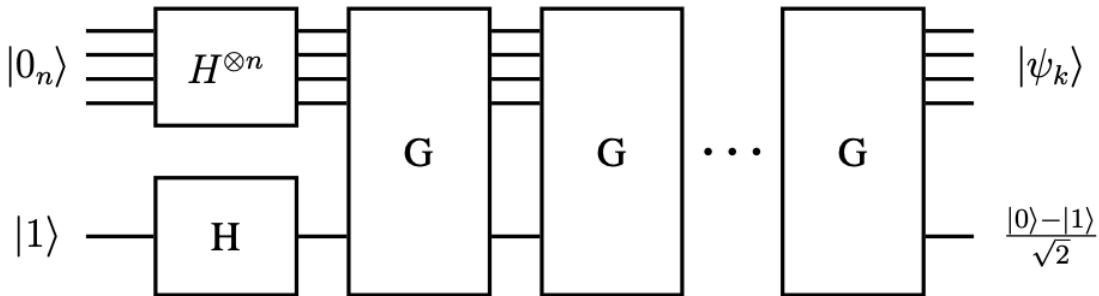


Figure 16: Circuit de l'algorithme Grover.

4.7.5.. Avantage sur le calcul classique:

L'algorithme de Grover présente une complexité en temps $O(\sqrt{N})$ car on répète environ $l \simeq \frac{\pi}{4}\sqrt{N}$ fois la porte G comme vu précédemment, ce qui est évidemment beaucoup plus que l'algorithme de la dichotomie sur une liste ordonnée, mais beaucoup moins que la recherche séquentielle qui est de complexité N . Par exemple, pour une liste de 1024 éléments, seulement environ 30 tests seraient nécessaires, tandis que pour une liste d'un milliard d'éléments, le nombre de tests requis serait d'environ 30000. Cependant, il est essentiel de souligner que l'algorithme est probabiliste, avec une probabilité d'erreur inférieure à N^{-4} .

Dans ce contexte, l'utilité d'un algorithme "imparfait" mais rapide dépend de la tolérance au risque et de la facilité de vérification du résultat. Par exemple, si 99% du temps l'algorithme trouve la bonne réponse, alors l'erreur occasionnelle peut être acceptable dans de nombreux cas pratiques. La possibilité de vérifier et de relancer l'algorithme en cas d'erreur le rend encore plus utile, même si le taux de succès n'est pas de 100%.

4.8.. Ordinateur quantique

À présent, nous allons faire tourner nos circuits quantiques. Pour cela, deux options s'offrent à nous. La première consiste à faire appel à de véritables ordinateurs quantiques qui exploitent directement les propriétés quantiques de certains éléments pour réaliser des calculs. On peut notamment faire tourner nos circuits sur les ordinateurs quantiques d'IBM grâce à leur plateforme IBM Quantum Composer [10] ou encore sur les ordinateurs quantiques d'IonQ [11]. La deuxième option consiste à simuler nos qubits à l'aide d'ordinateurs classiques, exploitant les mathématiques de la physique quantique que nous avons décrites précédemment. Pour se faire, les plateformes citées précédemment donnent accès à des simulateurs quantiques, mais nous pouvons aussi utiliser un super calculateur, comme Roméo [12] et son **QLM** (Quantum Learning Machine). Ces simulateurs permettent d'avoir accès à un plus grand nombre de qubits comparés aux véritables ordinateurs quantiques.

Pour mettre en œuvre un ordinateur quantique réel, plusieurs approches sont explorées. Les ions piégés [13], par exemple, représentent des qubits sous forme d'ions maintenus en suspension dans un piège électromagnétique. Ces ions sont ensuite manipulés à l'aide de lasers pour réaliser des opérations quantiques, formant ainsi des circuits qui exécutent des algorithmes à travers une séquence de portes quantiques. Une autre approche, celle des circuits supraconducteurs [14], fait appel à des matériaux refroidis à des températures proches du zéro absolu, permettant aux électrons de se comporter de manière quantique. Ces circuits supraconducteurs forment également des qubits qui peuvent être entrelacés et manipulés par des portes quantiques pour effectuer des calculs.



Figure 17: Ordinateur quantique supraconducteur d'IBM.

Chaque qubit dans ces architectures peut interagir avec d'autres qubits pour former des circuits plus complexes, réalisant ainsi des opérations qui seraient sinon impossibles pour les ordinateurs classiques. Ces interactions sont gouvernées par les portes quantiques, qui manipulent les états quantiques des qubits, analogues aux portes logiques des circuits électroniques classiques, mais avec la capacité de créer et de manipuler des superpositions et des intrications.

Ordinateur quantique supraconducteur

Les ordinateurs quantiques supraconducteurs [14], comme celui présenté sur la Figure 17, exploitent les propriétés quantiques des électrons dans des matériaux supraconducteurs [15]. Lorsque ces matériaux sont refroidis à des températures proches du zéro absolu, ils entrent dans un état où la résistance électrique est nulle. Dans cet état, les courants électriques, qui sont des flux d'électrons en mouvement, peuvent circuler sans perte d'énergie. Ainsi, l'architecture de ces ordinateurs intègre des éléments tels que l'or et le cuivre pour leur excellente conductivité et stabilité thermiques. La puce quantique, cœur de l'appareil, se trouve tout en bas de l'armature. Le tout est soigneusement placé dans une enceinte ultra-froide, souvent assimilée à un "frigo quantique", qui garantit des températures proches du zéro absolu et un environnement sous vide.

Le qubit, élément de base du calcul quantique dans ce contexte, est souvent représenté par ce qu'on appelle un qubit Josephson [16]. Ce qubit s'appuie sur la jonction Josephson, une fine interface entre deux supraconducteurs. Grâce aux propriétés quantiques, les électrons, regroupés en paires appelées paires de Cooper [17], peuvent "tunneler" à travers cette jonction. Cette action de tunneling est hautement prédictive et exploitable : elle permet de représenter les états quantiques $|0\rangle$ et $|1\rangle$, servant ainsi d'unité fondamentale de stockage et de traitement de l'information dans l'ordinateur quantique.

Ordinateur quantique à Ions

Les ordinateurs quantiques à ions piégés [13] représentent une technologie de pointe dans la recherche en informatique quantique. En utilisant des champs électromagnétiques précisément contrôlés, des ions comme ceux de l'ytterbium sont confinés dans une chaîne linéaire.

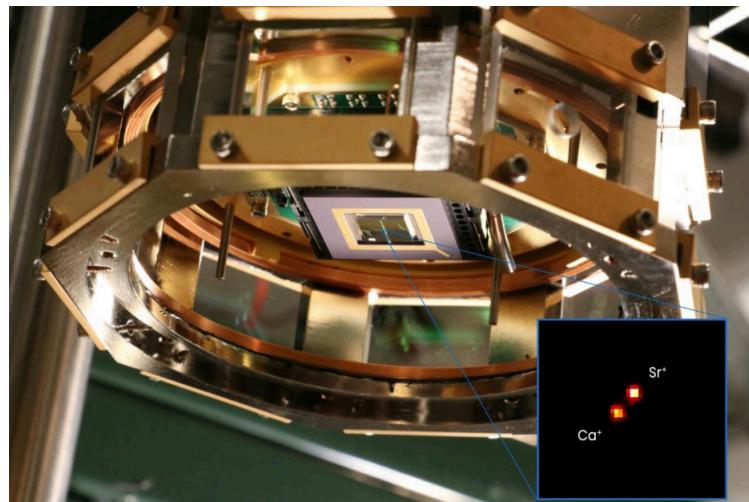


Figure 18: Ordinateur quantique à Ions du MIT Lincoln Laboratory researchers.

Les qubits sont formés en ionisant les atomes par l'extraction d'un électron à l'aide de lasers, ce qui prépare les ions à servir de supports d'information quantique. Ces ions piégés sont ensuite manipulés par des lasers pour effectuer des opérations logiques quantiques nécessaires à la réalisation de calculs complexes. Cette méthode est cruciale pour le développement potentiel d'ordinateurs quantiques à grande échelle, bien que des défis considérables subsistent en termes de scalabilité et de gestion des erreurs quantiques.

4.9.. QRAM

La Quantum Random Access Memory (QRAM) [18], [19] est une avancée théorique proposant une manière exponentiellement plus efficace d'accéder aux données stockées en mémoire, utilisant des qubits pour adresser une superposition d'états mémoires. Cela permettrait, en théorie, d'accéder à une vaste quantité de données simultanément, offrant un potentiel considérable pour l'accélération des algorithmes quantiques, notamment dans la reconnaissance de motifs, la cryptographie, et la recherche quantique sur bases de données classiques. La QRAM, grâce à son architecture *bucket-brigade*, réduit le nombre de commutations nécessaires pour un appel mémoire de $O(N)$ à $O(\log N)$, promettant des gains substantiels en termes de vitesse et d'économie d'énergie. Cependant, malgré ses avantages théoriques impressionnantes, la réalisation pratique de la QRAM reste un défi majeur, avec des obstacles techniques et des questions ouvertes quant à sa faisabilité et son efficacité réelle dans les systèmes quantiques actuels et futurs.

4.10.. Le Q-Means: La Rencontre du Quantique et du Clustering

Les concepts fondamentaux de la physique et de l'informatique quantique trouvent des applications pratiques notamment dans le domaine du machine learning. L'algorithme Q-Means en est un exemple. Inspiré de sa contrepartie classique le δ -K-Means, expliqué dans la [Section 2](#), l'algorithme Q-Means adapte plusieurs de ses étapes clés pour bénéficier des avantages du quantique.

À un niveau élevé, l'algorithme Q-Means suit la même séquence d'étapes que le K-Means classique. Toutefois, il utilise des sous-routines quantiques pour des tâches spécifiques comme l'estimation des distances et la recherche d'une valeur minimale parmi un ensemble d'éléments pour obtenir de nouveaux centroïdes sous forme d'états quantiques. Par ailleurs, Q-Means vient avec son initialisation quantique des centroïdes, appelée Q-Means++ [1], qui est l'équivalent quantique de l'efficace K-Means++ [2] détaillée dans la [Section 3](#). Contrairement à la version classique où chaque point de données est attribué séquentiellement à des clusters, lors du Q-Means, cette affectation est réalisée en superposition. Enfin, les étapes de mise à jour des centroïdes sont également réalisées de manière quantique, et le processus se répète jusqu'à convergence.

Dans cet article, nous tiendrons compte des contraintes liées à l'absence de la QRAM et du nombre de qubits disponibles sur les ordinateurs quantiques actuels. La section suivante présente les différentes briques qui seront utilisées dans la création de cet algorithme QMeans.

5.. PROCÉDURES QUANTIQUES POUR L'ALGORITHME Q-MEANS

5.1.. Estimation quantique de la distance euclidienne entre des points

Sur un ordinateur classique, les distances euclidiennes sont directement calculables. Cependant, sur un ordinateur quantique, cette tâche est compliquée à cause de la nature probabiliste des qubits. Malgré cette complexité, il est essentiel d'avoir une manière d'estimer les distances entre les points. Pour cela, on peut par exemple utiliser le produit scalaire, comme introduit par Stephen DiAdamo et Al. [20], comme indicateur de leur distance.

Le principe repose sur le fait que nous n'avons pas besoin de connaitre la distance exacte entre les points, mais plutôt d'avoir une estimation qui pourra permettre de comparer plusieurs valeurs de distances. Ainsi, le produit scalaire, bien que non proportionnel à la distance, est positivement corrélé avec celle-ci, ce qui permet par exemple de déterminer de manière efficace quel point est le plus proche d'un autre point donné.

5.1.1.. Relation entre le produit scalaire et la distance:

Imaginons que nous avons un qubit auxiliaire initialisé à l'état $|0\rangle$ et un état quantique $|\psi\rangle = |x\rangle|y\rangle$ qui représente les deux vecteurs normalisés dont nous voulons estimer la distance.

Voici les étapes permettant de mettre en évidence la relation entre la probabilité de mesurer le qubit auxiliaire dans l'état $|1\rangle$ (notée $P(1_{\text{aux}})$) et le produit scalaire des vecteurs:

- **Définition de $|\psi\rangle$ et Application de la Porte Hadamard :** Commençons par un qubit auxiliaire initialisé à $|0\rangle$ et un état quantique $|\psi\rangle$, qui stockent les vecteurs normalisés dont nous voulons estimer la distance :

$$|\psi\rangle = |x\rangle|y\rangle \quad (55)$$

Après l'application d'une porte Hadamard au qubit auxiliaire, et en utilisant la porte de SWAP contrôlée (C-SWAP) sur le qubit auxiliaire, l'état devient :

$$\frac{1}{\sqrt{2}}(|0\rangle|\psi\rangle + |1\rangle F(|\psi\rangle)) \quad (56)$$

où $F(|\psi\rangle) = |y\rangle|x\rangle$.

- **Manipulation avec une Seconde Porte Hadamard :** Une autre porte Hadamard est appliquée au qubit auxiliaire, résultant en :

$$\frac{1}{2}(|0\rangle|(I + F)\psi\rangle + |1\rangle|(I - F)\psi\rangle) \quad (57)$$

où I est l'opérateur identité.

- **Introduction de la probabilité que le qubit auxiliaire soit à $|1\rangle$:** On peut substituer $(I - F)$ et $(I + F)$, ce qui donne :

$$(|0\rangle|A_0\psi\rangle + |1\rangle|A_1\psi\rangle) \quad (58)$$

où $A_0 = \frac{I+F}{2}$ et $A_1 = \frac{I-F}{2}$

À présent, en utilisant le concept de valeur moyenne [21] décrite dans la Section 4.5.3 on peut exprimer la probabilité que le qubit auxiliaire soit mesuré dans l'état $|1\rangle$ avec un produit interne :

$$P(1_{\text{aux}}) = \langle \psi | A_1 | \psi \rangle \quad (59)$$

- **Simplifications**: On simplifie par substitution avec I et F :

$$P(I_{\text{aux}}) = \left\langle xy \left| \frac{I - F}{2} \right| xy \right\rangle \quad (60)$$

En développant et sachant que $\langle xy, xy \rangle = 1$ et $\langle xy, yx \rangle = \langle x, y \rangle^2$, on obtient :

$$P(1_{\text{aux}}) = \frac{1}{2} - \frac{1}{2} \langle x, y \rangle^2 \quad (61)$$

Finalement, nous obtenons une relation entre la probabilité de mesurer le qubit auxiliaire dans l'état $|1\rangle$ (notée $P(1_{\text{aux}})$) et le produit scalaire des vecteurs.

Théorème 5.1.1.1: La probabilité de mesurer le qubit auxiliaire dans l'état $|1\rangle$ (c'est à dire $P(1_{\text{aux}})$) est positivement corrélée avec la distance entre les vecteurs $|x\rangle$ et $|y\rangle$. Ce qui permet de comparer les distances sans les calculer explicitement.

Démonstration: Montrons que la probabilité $P(1_{\text{aux}})$ calculée précédemment est positivement corrélée avec la distance euclidienne entre $|x\rangle$ et $|y\rangle$.

Le carré de la distance euclidienne entre deux vecteurs est définie comme :

$$d^2 = \|x - y\|^2 \quad (62)$$

En développant la soustraction vectorielle, comme les vecteurs sont normalisés on obtient:

$$d^2 = \langle x - y, x - y \rangle = 2 - 2\langle x, y \rangle \quad (63)$$

Si vous regardez de près, $P(1_{\text{aux}})$ et le carré de la distance euclidienne ont des formes qui sont similaires en fonction du produit interne $\langle x, y \rangle$. En particulier, $P(1_{\text{aux}})$ est positivement corrélé avec d^2 car à mesure que la distance entre les vecteurs augmente (c'est-à-dire qu'ils deviennent plus orthogonaux), le produit interne $\langle x, y \rangle$ diminue, et par conséquent $P(1_{\text{aux}})$ augmente.

Il est important de noter que bien que nous comparons le carré de la distance euclidienne d^2 plutôt que la distance euclidienne d elle-même, cela ne pose pas de problème car la fonction carré est monotone pour les distances euclidiennes positives. Ainsi, une augmentation de d^2 indique toujours une augmentation de d , et vice versa. \square

5.1.2.. Encodage des coordonnées vectorielles en qubits:

Pour appliquer l'algorithme quantique, il est essentiel d'encoder les coordonnées des points en qubits. Ceci est réalisé en convertissant les coordonnées rectangulaires classiques en coordonnées sphériques, adaptées à une représentation sur la sphère de Bloch:

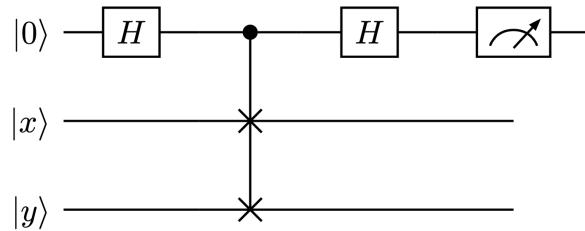
- Commencez avec un qubit initialisé à $|0\rangle$.
- Appliquez une porte Hadamard H pour orienter le qubit le long de l'axe X de la sphère de Bloch. (Nous utiliserons plutôt $\lambda = 0$ dans une porte U plutôt qu'une porte H)
- Les angles φ et θ sont utilisés pour représenter les valeurs des deux caractéristiques du point. Ici, θ peut varier entre 0 et π radians, tandis que φ est limité entre 0 et π pour éviter des problèmes de représentation sur la sphère de Bloch.
- Les relations suivantes permettent de mapper les caractéristiques du point vers les angles φ et θ :

$$\varphi = (d_0 + 1)\frac{\pi}{2} \quad \theta = (d_1 + 1)\frac{\pi}{2} \quad (64)$$

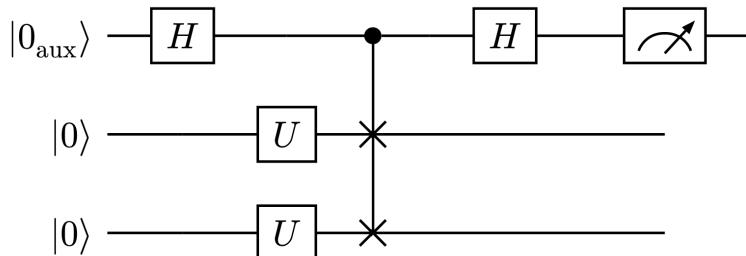
Où d_0 et d_1 correspondent aux valeurs des deux caractéristiques du point.

5.1.3.. Circuit quantique associé:

Le circuit quantique théorique correspondant à cette étape d'estimation de la distance est le suivant :



Le qubit $|0\rangle$ étant le qubit auxiliaire et les qubits $|x\rangle$ et $|y\rangle$ les qubits représentant les deux vecteurs dont nous voulons estimer leur *distance*. Cependant, en considérant la Section 5.1.2 précédente, on peut détailler un peu plus le circuit, ce qui donne :



Où $|0_{\text{aux}}\rangle$ est le qubit auxiliaire, et les deux qubits $|x\rangle$ et $|y\rangle$ sont créés à partir d'un qubit $|0\rangle$, d'une porte U avec comme paramètres θ et φ calculés dans la Section 5.1.2 et λ avec comme valeur 0 pour aligner le qubit le long de l'axe X de la sphère de Bloch. La mesure sur le qubit auxiliaire permet de calculer la probabilité de trouver $|0_{\text{aux}}\rangle$ dans l'état $|1\rangle$ après n circuits lancés.

Cette porte U est définie par:

$$U(\theta, \varphi, \lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda} \sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi} \sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)} \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \quad (65)$$

De manière plus théorique, on peut exprimer ce circuit qui estime la distance, par l'opérateur que l'on note \hat{K} et que l'on définit par :

$$\hat{K} = (\hat{H} \otimes \hat{I} \otimes \hat{I}) \cdot \text{C-SWAP}_{\text{aux},(x,y)} \cdot (\hat{H} \otimes \hat{U}(\theta_x, \varphi_x, \lambda_x) \otimes \hat{U}(\theta_y, \varphi_y, \lambda_y)) \quad (66)$$

Où

- θ_x, φ_x et λ_x proviennent de l'encodage du premier point
- θ_y, φ_y et λ_y proviennent de l'encodage du deuxième point
- $\text{C-SWAP}_{\text{aux},(x,y)}$ est la porte SWAP entre le qubit **x** et **y** (les deux points) contrôlée par le qubit **aux** (le qubit auxiliaire)

Que l'on utilise en calculant, après avoir au préalable calculé les valeurs des paramètres de la porte U :

$$|\psi\rangle = \hat{K} \cdot |000\rangle \quad (67)$$

On obtient le résultat de la pseudo distance en mesurant le qubit auxiliaire, qui est le premier qubit, et en comptant le nombre de fois où il est mesuré dans l'état $|1\rangle$. Pour mesurer maintenant la distance entre 1 point et n autres points on va mettre en parallèle cet opérateur (donc ce circuit) pour effectuer l'algorithme sur les n points en même temps. Voici la forme du circuit :

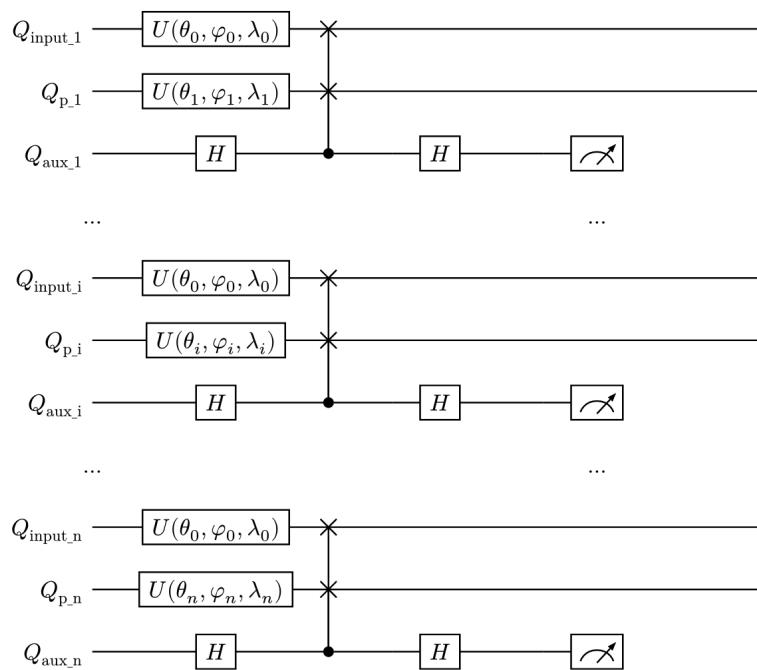


Figure 21: Circuit quantique pour le calcul des distances avec n points.

Nous avons n registres de 3 qubits, chacun contient un qubit qui correspond à la duplication du point dont il faut trouver la distance avec les n autres, un qubit représentant l'un des n points et un qubit auxiliaire. On peut exprimer ce circuit comme un opérateur \hat{K}_n à l'aide de l'opérateur \hat{K} défini précédemment, tel que :

$$\hat{K}_n = \bigotimes_{i=0}^n \hat{K}_i \quad (68)$$

Où \hat{K}_i est défini par :

$$\hat{K}_i = (\hat{H} \otimes \hat{I} \otimes \hat{I}) \cdot \text{C-SWAP}_{\text{aux_i},(\text{x},\text{p_i})} \cdot (\hat{H} \otimes \hat{U}(\theta_0, \varphi_0, \lambda_0) \otimes \hat{U}(\theta_i, \varphi_i, \lambda_i)) \quad (69)$$

Où

- θ_0, φ_0 et λ_0 proviennent de l'encodage du premier point
- θ_i, φ_i et λ_i proviennent de l'encodage du i -ème point parmi les n
- C-SWAP_{aux_i,(x,p_i)} est la porte SWAP entre le qubit **x** et **p_i** (le point et le i -ème point) contrôlée par le qubit **aux_i** (le i -ème qubit auxiliaire)

Dont le circuit correspondant est :

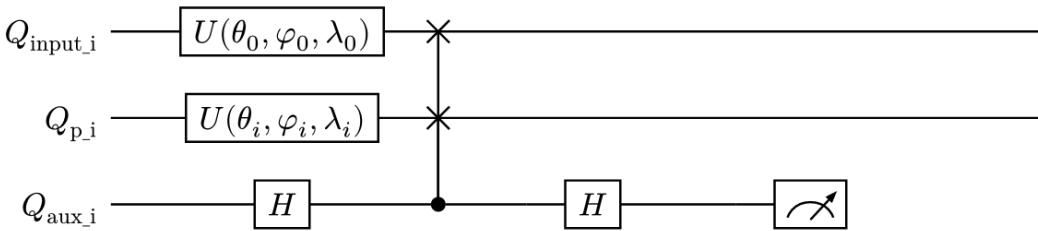


Figure 22: i -ème composante du circuit de calcul de distances.

Le calcul des distances entre 1 donnée et n points nécessite donc $3n$ qubits.

Après la mesure des n qubits auxiliaires, chaque bit des états finaux représente un des n points spécifiques (celui qui est impliqué dans la porte C-SWAP dans le circuit présenté dans la [Figure 23](#)). Pour interpréter les résultats, nous comptons les occurrences pour chaque position du bit où un 1 apparaît, c'est à dire le bit à l'index i .

Ainsi, l'index du point associé à un nombre réduit d'occurrences de 1 est considéré comme le plus probable d'être le plus proche du point en question, en raison de la manière dont les distances sont mesurées dans ce contexte. En effet, comme démontré dans la [Section 5.1.1](#), la probabilité de mesurer le qubit auxiliaire dans l'état $|1\rangle$ est positivement corrélée avec la distance entre le premier point et le deuxième. Le qubit auxiliaire étant représenté par le bit à la position i dans la sortie du circuit.

Par exemple avec $n = 5$ points on obtient le circuit de la Figure 23.

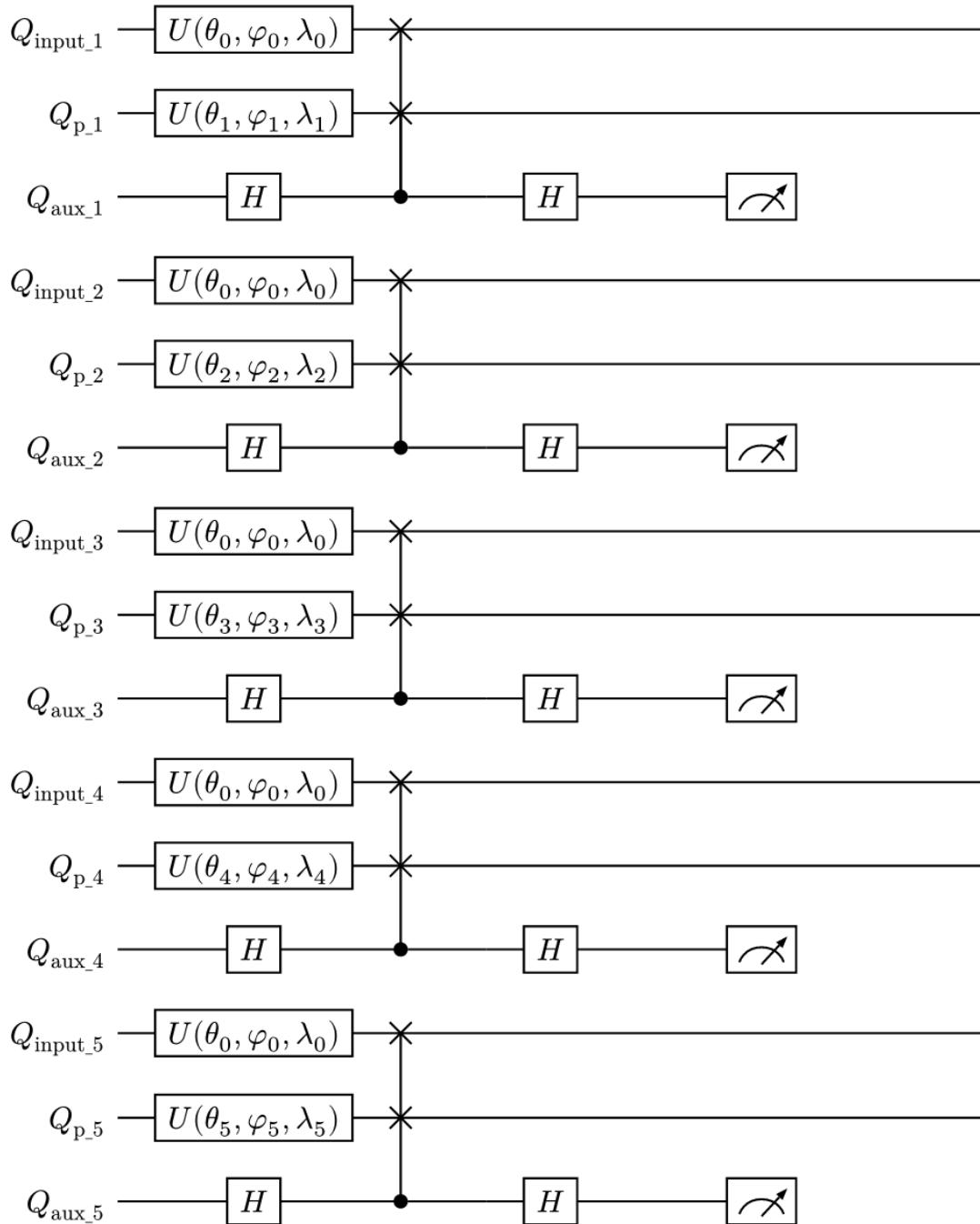


Figure 23: Circuit quantique pour le calcul des distances avec 5 clusters.

Ainsi, les qubits Q_{input_i} sont la duplication par **5** de la donnée dont il faut trouver la distance avec les points. Les qubits Q_{p_i} sont les **5** points. Les qubits Q_{aux_i} sont les qubits auxiliaires $|0_{\text{aux}}\rangle$. À la fin, on mesure l'état des **5** qubits auxiliaires.

On obtient les résultats illustrés dans la [Figure 24](#). Nous utiliserons les simulateurs d'ordinateurs quantiques d'IBM (celui nommé *ibmq_qasm_simulator*). Ici, le calcul dure 343 millisecondes, en ayant effectué 4096 tours.

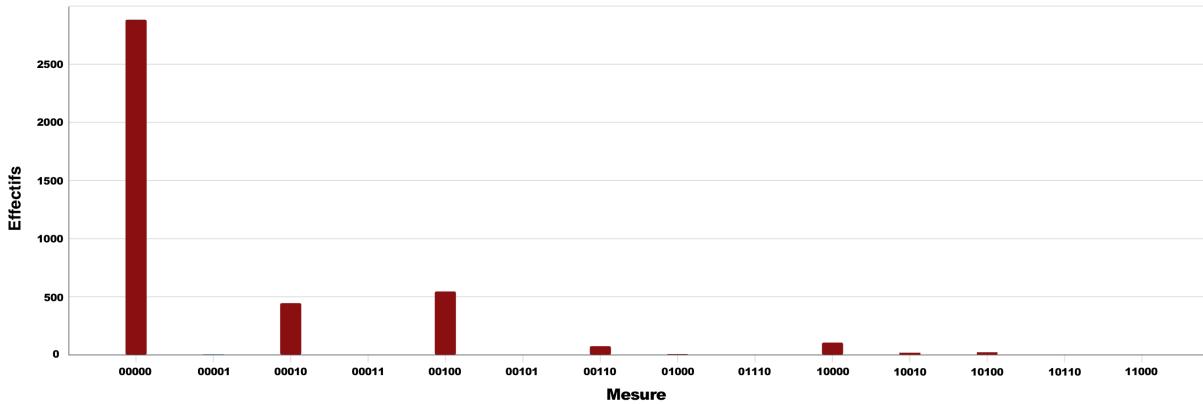


Figure 24: Histogramme de sortie du circuit.

Après simulation, nous obtenons les fréquences des différents états quantiques sur 5 bits, et nous comptons le nombre de 1 qui apparaît pour chaque position. On remarquera que l'index associé au minimum de valeur correspond au point le plus probable d'être le plus proche du point en question.

5.1.4.. **Exemple à la main:**

On prend le point X de coordonnées $(x_0, y_0) = (0.1, 0.2)$, Ainsi que les points p_1, p_2 et p_3 de coordonnées respectives $(x_1, y_1) = (0.1, 0.2)$, $(x_2, y_2) = (0.3, 0.4)$ et $(x_3, y_3) = (0.5, 0.6)$. On commence par les encoder en qubits suivant la [Section 5.1.2](#).

Pour $X(x_0, y_0) = (0.1, 0.2)$:

$$\varphi_X = (x_0 + 1)\frac{\pi}{2} = (0.1 + 1)\frac{\pi}{2} \quad \theta_X = (y_0 + 1)\frac{\pi}{2} = (0.2 + 1)\frac{\pi}{2} \quad (70)$$

Pour $p_1(x_1, y_1) = (0.1, 0.2)$:

$$\varphi_{C1} = (x_1 + 1)\frac{\pi}{2} = (0.1 + 1)\frac{\pi}{2} \quad \theta_{p1} = (y_1 + 1)\frac{\pi}{2} = (0.2 + 1)\frac{\pi}{2} \quad (71)$$

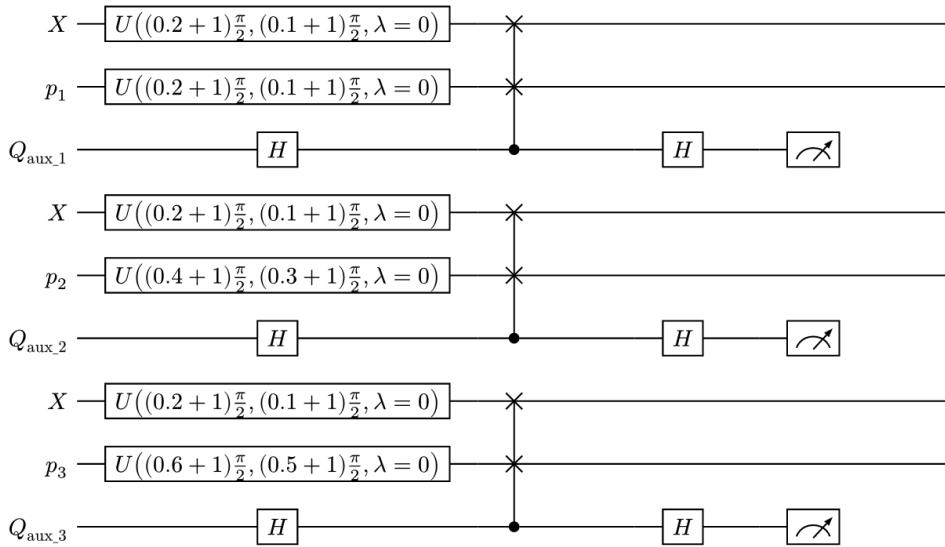
Pour $p_2(x_2, y_2) = (0.3, 0.4)$:

$$\varphi_{C2} = (x_2 + 1)\frac{\pi}{2} = (0.3 + 1)\frac{\pi}{2} \quad \theta_{p2} = (y_2 + 1)\frac{\pi}{2} = (0.4 + 1)\frac{\pi}{2} \quad (72)$$

Pour $p_3(x_3, y_3) = (0.5, 0.6)$:

$$\varphi_{C3} = (x_3 + 1)\frac{\pi}{2} = (0.5 + 1)\frac{\pi}{2} \quad \theta_{p3} = (y_3 + 1)\frac{\pi}{2} = (0.6 + 1)\frac{\pi}{2} \quad (73)$$

On obtient alors le circuit suivant:



En lançant 4096 fois ce circuit on obtient les fréquences suivantes: {'001': 322, '011': 4, '000': 3689, '010': 81}. Comme expliqué précédemment, il faut compter le nombre d'apparitions, pour chaque position de 1. Ce qui nous donne le tableau de fréquences suivant : [0, 85, 326]. C'est à dire que le premier qubit auxiliaire a été mesuré à $|1\rangle$ aucune fois, le deuxième 85 fois et le troisième 326 fois. Comme la probabilité que chaque qubit auxiliaire soit dans l'état $|1\rangle$ est proportionnelle à la distance entre les deux points impliqués dans le SWAP on cherche alors la plus petite valeur. Ici, c'est 0 le minimum, ce qui signifie que le premier point p_1 est le plus proche du point X . (Ce qui est vrai, en regardant les coordonnées des points). Ce calcul du minimum sur cette liste en particulier est l'objet de l'affectation des clusters dans la [Section 5.3](#).

À partir de là, on peut estimer le pourcentage de bonne réponse en testant ce circuit sur plusieurs données aléatoires. On observe par exemple qu'avec $n = 5$ points, en faisant tourner notre algorithme 200 fois et avec 4096 shots, on obtient 80% de bonnes réponses. C'est à dire qu'on arrive à trouver la bonne valeur minimale, parmi les 5, avec une probabilité de 80%. Avec $n = 3$ points on monte à 90% de réussite. Cet effet aléatoire sur le calcul des distances est similaire à son homologue classique le δ -k-means.

5.1.5.. Avantage sur le calcul classique:

Alors que sur un ordinateur classique, il faudrait calculer explicitement la distance euclidienne pour chaque point, sur un ordinateur quantique, nous n'avons qu'à mesurer la probabilité $P(1_{\text{aux}})$. C'est beaucoup plus efficace, surtout lorsque vous avez un grand nombre de points à traiter. En fin de compte, même si l'algorithme quantique ne nous donne pas la distance exacte, il nous donne suffisamment d'informations pour déterminer le point le plus proche, ce qui est l'objectif principal dans l'algorithme Q-Means.

5.2.. Comparer deux entiers

Pour comparer deux entiers nous utiliserons l'algorithme **Quantum bit string comparator** [22] introduit en 2007 par David Sena Oliveira et Rubens Ramos. On commencera par comparer deux bits puis nous étendrons cela à la comparaison de deux entiers sur n bits. Au final, nous pourrons savoir, à partir de deux entiers a et b , si $a < b$ ou $a \geq b$.

5.2.1.. Encodage des entiers en qubits:

Afin de pouvoir comparer deux entiers à l'aide d'un circuit quantique, il est essentiel d'encoder ces nombres en qubits. Pour cela, nous transformerons les entiers en leur valeur binaire.

Ainsi, l'encodage de nombres dans un circuit quantique nécessite un nombre de qubits correspondant à la taille des nombres en question. Par exemple, un qubit peut encoder deux valeurs (0 et 1), tandis que deux qubits peuvent encoder quatre valeurs ($00 = 0$, $01 = 1$, $10 = 2$, $11 = 3$). Au final, n qubits permettent d'encoder 2^n nombres.

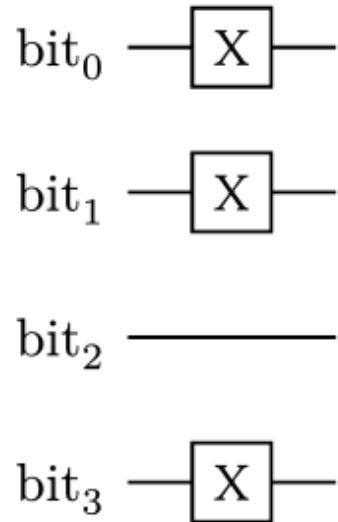


Figure 26: Encodage de l'entier 13, '1101' en binaire.

En parcourant l'écriture binaire de l'entier, si le bit est 1, une porte X (porte NOT) est appliquée au qubit, le changeant de l'état $|0\rangle$ à l'état $|1\rangle$. Si le bit est 0, aucune action n'est entreprise, et le qubit reste dans l'état $|0\rangle$. Ainsi, on traduit un bit classique en un état quantique.

5.2.2.. Comparer deux entiers:

À présent, le but est de comparer deux entiers qui sont encodés dans le circuit quantique grâce à la méthode décrite précédemment. La comparaison des entiers est effectuée bit à bit et utilise des qubits auxiliaires pour indiquer quel bit est le plus grand. La comparaison commence par les bits les plus significatifs des deux entiers. Si ces bits sont égaux, le processus se poursuit avec les bits suivants. Si les bits sont différents, la fonction de comparaison bit à bit détermine quel entier est le plus grand.

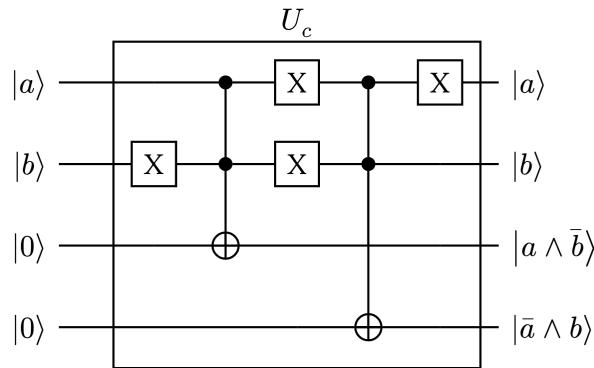


Figure 27: Opérateur pour comparer deux bits.

La Figure 27 présente le circuit U_c permettant de déterminer quel bit est le plus grand. Il nécessite deux qubits $|a\rangle$ et $|b\rangle$ pour représenter les deux bits et deux qubits auxiliaires initialisés à $|0\rangle$.

On prépare les états $|a\rangle$ et $|b\rangle$ suivant la Section 5.2.1 précédente. Puis, une porte NOT multi-controlée est appliquée sur le premier qubit auxiliaire avec la paire $|a\rangle$ et $|b\rangle$ comme contrôle. Ce premier qubit auxiliaire est donc dans l'état $|a \wedge \bar{b}\rangle$. Ensuite, une deuxième porte NOT multi-controlée est appliquée sur le deuxième qubit auxiliaire avec la paire $|a\rangle$ et $|b\rangle$ comme contrôle, $|a\rangle$ ayant subit une porte NOT. Ainsi, le deuxième qubit auxiliaire contiendra l'état $|\bar{a} \wedge b\rangle$.

Ces états peuvent être explicités suivant les différentes valeurs de $|a\rangle$ et $|b\rangle$ à l'aide d'une table de vérité:

| $ a\rangle$ | $ b\rangle$ | $ a \wedge \bar{b}\rangle$ | $ \bar{a} \wedge b\rangle$ |
|-------------|-------------|----------------------------|----------------------------|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

On peut en conclure que si les qubits auxiliaires sont dans l'état $|01\rangle$ alors $a < b$, sinon $a \geq b$.

On définit de plus l'opérateur U_c^{-1} qui permet de décalculer les qubits utilisés par l'opérateur U_c . On construit alors l'opérateur suivant:

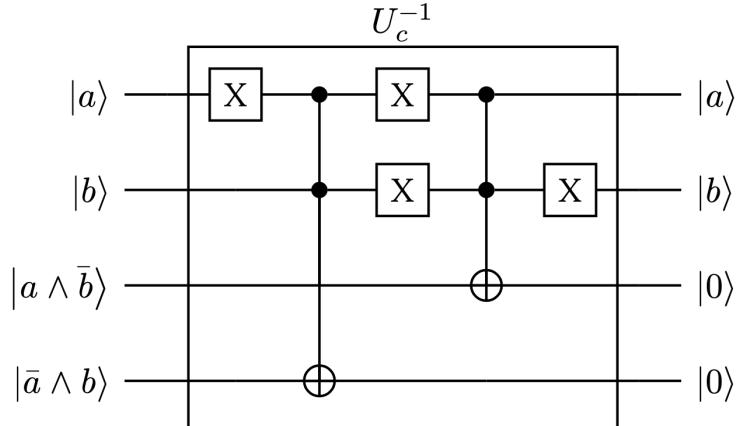


Figure 28: Opérateur pour décalculer les qubits de l'opérateur de comparaison.

Au final, voici le circuit permettant de déterminer quel bit est le plus grand entre $|a_0\rangle$ et $|b_0\rangle$. Le dernier qubit est le qubit qui contient le résultat final de la comparaison. (comme précédemment, les qubits $|a_0\rangle$ et $|b_0\rangle$ sont préparés suivant la [Section 5.2.1](#).

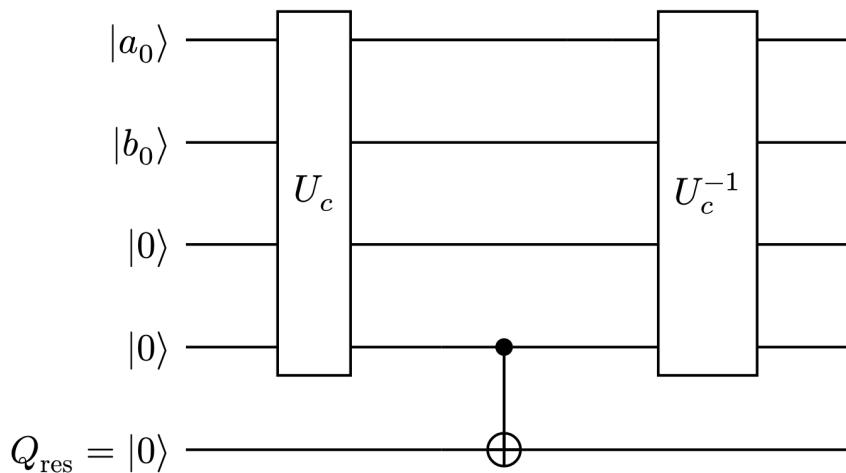


Figure 29: Circuit complet pour comparer deux bits $|a_0\rangle$ et $|b_0\rangle$.

Ainsi, en suivant la table de vérité précédente et en remarquant que la comparaison est déterminée par les qubits auxiliaires de l'opérateur U_c , on peut interpréter les résultats de la manière suivante:

$$\begin{cases} a_0 < b_0 & \text{si } Q_{\text{res}} = |1\rangle \\ a_0 \geq b_0 & \text{si } Q_{\text{res}} = |0\rangle \end{cases} \quad (74)$$

Pour comparer plus de bits à la fois on va faire en sorte que l'information de comparaison générée par l'opérateur U_c soit propagée à travers le circuit de manière conditionnelle, des bits les plus significatifs aux bits les moins significatifs. Pour cela, on utilise la porte multi-controlée X (MCX). Cette porte permet d'appliquer une opération sur un qubit cible seulement si certaines conditions sur les qubits de contrôle sont satisfaites. Dans ce contexte, elle permet de propager l'information sur les résultats des comparaisons bit à bit précédentes à travers le circuit. Dès qu'on trouve que l'un des bits est plus grand que l'autre alors on peut conclure et dire quel entier est le plus grand. Pour commencer, on va donner le circuit pour comparer deux entiers sur 2 bits.

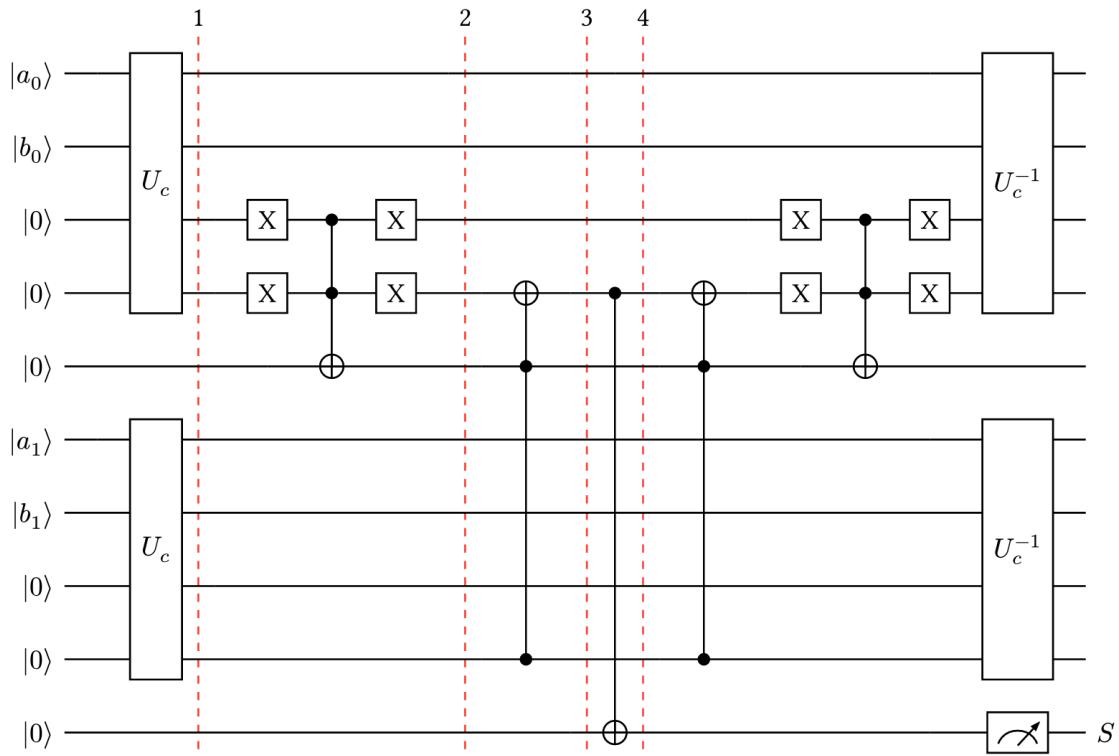


Figure 30: Circuit pour comparer deux entiers a et b sur 2 bits.

Le circuit de comparaison se décompose en plusieurs parties. On utilisera les lignes en pointillées rouges pour les situer. On rappelle que les qubits $|a_i\rangle$ et $|b_i\rangle$ sont préparés suivant la Section 5.2.1.

- Les portes U_c avant la ligne 1 permettent d'effectuer les comparaisons bit à bit.
- Entre la ligne 1 et la ligne 2, on applique une porte X multi-controlée pour modifier le qubit auxiliaire pour chaque paire de qubits (sauf la dernière) quand les qubits auxiliaires de la porte U_c sont à $|00\rangle$.
- Entre la ligne 2 et la ligne 3 on fait remonter les résultats de comparaison des qubits les moins significatifs vers les qubits les plus significatifs.
- Entre la ligne 3 et 4 on met à jour le qubit de résultat final.
- Après la ligne 4 on décalcule tous les qubits utilisés, sauf le qubit contenant le résultat S .

Pour obtenir les résultats, comme précédemment, on mesure le qubit auxiliaire S et on observe l'état de ce qubit sur plusieurs shots. L'état le plus probable permet de conclure sur la comparaison des deux entiers. Si $S = |0\rangle$ alors $a \geq b$ et sinon si $S = |1\rangle$ alors $a < b$.

Dans la [Figure 30](#), nous encodons deux entiers a et b . L'entier a s'écrit de manière binaire comme a_0a_1 et b s'écrit en b_0b_1 , ainsi le bit significatif est le bit d'indice 0. On compare donc en premier les bits significatifs des deux entiers, si on peut en déduire à cet endroit lequel est le plus petit les comparaisons suivantes n'auront pas d'impact, sinon on continue avec le bit suivant dans chaque entier. Au final, on mesure une chaîne de 1 bit S . On peut généraliser ce circuit pour la comparaison de 2 entiers sur n bits en gardant la même logique pour le résultat, c'est à dire que si $S = |0\rangle$ alors $a \geq b$ et sinon si $S = |1\rangle$ alors $a < b$.

On obtient alors le circuit suivant:

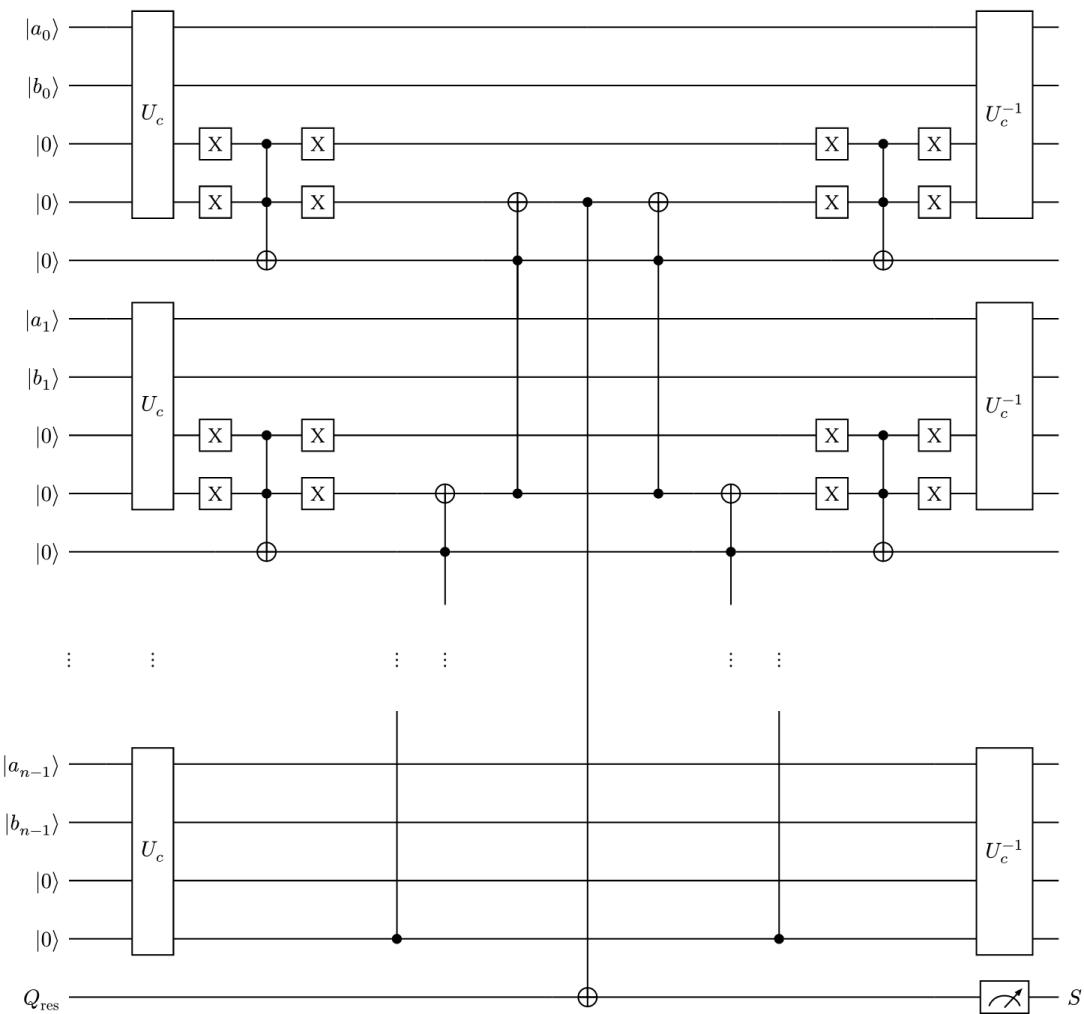


Figure 31: Circuit pour comparer deux entiers a et b sur n bits.

Pour comparer 2 entiers sur n bits il faudra ainsi $5n$ qubits.

5.2.3.. Exemples à la main:

1. On prend ici $a = 6$ et $b = 5$. On a donc $a = a_0a_1a_2 = 110$ et $b = b_0b_1b_2 = 101$. On obtient $S = |0\rangle$ ce qui veut dire que a est supérieur ou égal à b .

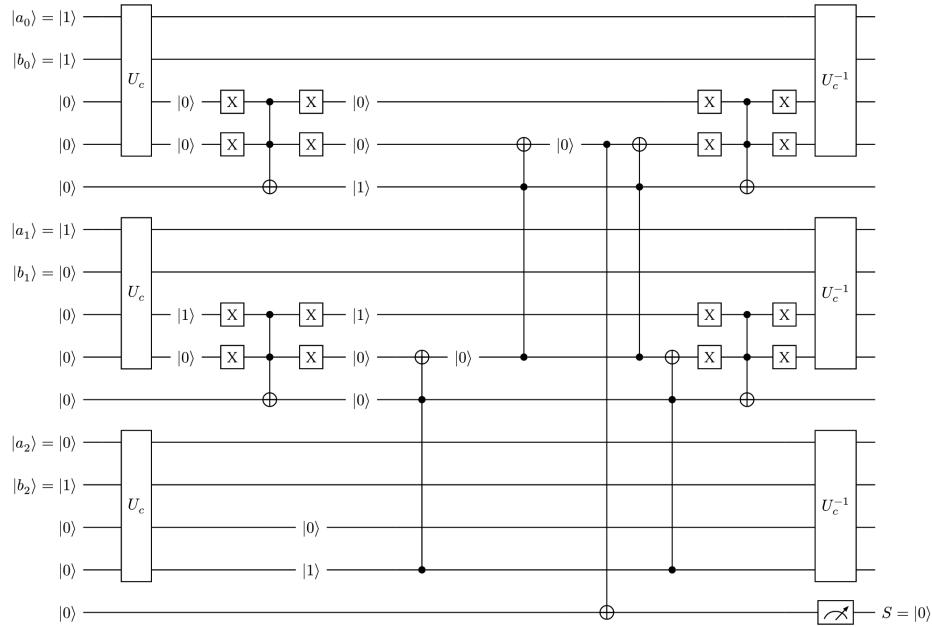


Figure 32: Exemple 1 avec $a = 6$ et $b = 5$

2. On prend ici $a = 4$ et $b = 6$. On a donc $a = a_0a_1a_2 = 100$ et $b = b_0b_1b_2 = 110$. On obtient $S = |1\rangle$ ce qui veut dire que a est inférieur strictement à b .

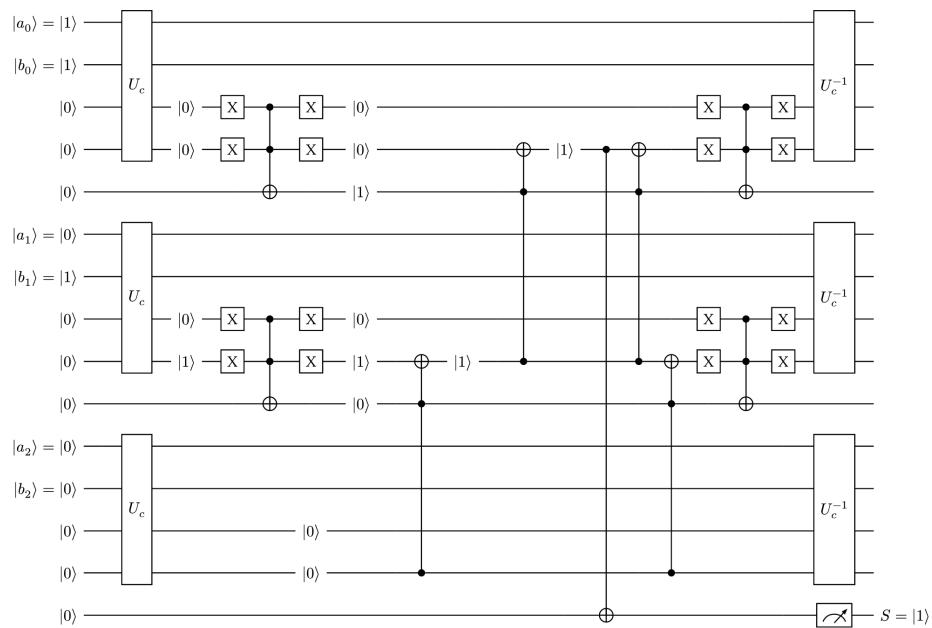


Figure 33: Exemple 2 avec $a = 4$ et $b = 6$

5.3.. Calcul quantique du minimum d'une liste d'entiers

Pour trouver le minimum d'une liste d'entiers, il existe plusieurs algorithmes quantiques comme par exemple celui de **Dürr-Hoyer** [23] ou encore l'algorithme **Quantum Minimum Search** [24], tout deux basés sur l'algorithme de Grover (voir [Section 4.7](#)). Dans cet article nous utiliserons le circuit final construit dans la [Section 5.2](#) précédente pour créer un Oracle afin de nous rapprocher de l'algorithme de **Dürr-Hoyer**.

Les étapes de l'algorithme:

1. **Choix de la valeur de référence y_i :** Si $i = 0$ alors on choisit y_0 aléatoire dans la liste, sinon on utilise la valeur de sortie à l'itération précédente. La représentation en binaire de cette valeur nécessite n qubits et sera la valeur de référence pour les comparaisons dans l'Oracle de comparaison des entiers.
2. **Initialisation de la machine quantique:** On initialise une machine quantique dans l'état

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} |x\rangle |0\rangle^{\otimes n} |- \rangle \quad (75)$$

Cet état est obtenu par l'opération $H^{\otimes n} I^{\otimes n}$ qui crée une superposition sur le premier registre (les $|x\rangle$) et garde le seconde registre dans l'état $|0\rangle^{\otimes n}$. L'état $|-\rangle$ pour le qubit auxiliaire, est obtenu en appliquant une porte Pauli-X suivi d'une porte d'Hadamard à un qubit $|0\rangle$. Cette préparation permet d'exploiter l'effet de phase kickbase expliqué dans la [Section 4.6](#).

3. **Préparation des états superposés:** La superposition crée 2^n états, mais tous ces états ne correspondent pas forcement aux éléments de la liste. Ainsi, on va amplifier les états présents dans la liste en utilisant l'opérateur de Grover appliqué à plusieurs solutions (Le nombre de valeurs dans la liste).
4. **Initialisation de la valeur y_i dans la machine quantique:** On initialise les n qubits $|0\rangle$ pour qu'ils représentent la valeur y_0 sur n bits, suivant la logique décrite dans la [Section 5.2.1](#). On obtient donc l'état

$$|\psi_1\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n} |x\rangle |y_i\rangle |- \rangle \quad (76)$$

5. **Application de l'Oracle de comparaison et de la diffusion:** L'Oracle de comparaison des entiers est appliqué pour marquer les états correspondant aux entiers inférieurs à y_i . Si un état satisfait cette condition, l'Oracle inversera la phase de cet état grâce au mécanisme de phase kickback expliqué dans la [Section 4.6](#). C'est à dire que l'on appliquera une porte Pauli-Z sur le qubit auxiliaire.

L'opérateur de diffusion est ensuite appliqué pour amplifier les probabilités des états marqués, augmentant ainsi la chance de mesurer un état qui représente un entier inférieur à y_i .

7. **Mesure et logique d'itération:** On effectue des mesures sur les n premiers qubits. Les états les plus fréquemment observés sont les états qui représentent les entiers inférieurs à y_i .

Si la valeur avec la plus grande probabilité est bien dans la liste et que le nombre maximum d'itération \sqrt{N} n'est pas atteint, elle devient la nouvelle référence y_i . Si elle n'est pas dans la liste alors soit on choisi une nouvelle valeur y_0 si c'est la première itération, sinon on conclut et on sort de la boucle en donnant comme réponse la dernière valeur de y_i appartenant à L .

Dans la version de l'algorithme de Dürr-Hoyer [23], la QRAM est utilisée (voir [Section 4.9](#)) pour dynamiquement sélectionner et changer la valeur de référence y_i . Cependant, pour pouvoir implémenter et exécuter notre algorithme, nous ne pouvons pas utiliser la QRAM, comme expliqué dans la [Section 4.9](#). C'est en cela que notre algorithme diffère de celui de Dürr-Hoyer. Nous devons lancer plusieurs fois le circuit en changeant la valeur de référence y_i .

L'algorithme de recherche du minimum d'une liste d'entiers codés sur n bits nécessite $5n$ qubits. En effet, il effectue des itérations du circuit de comparaison de deux entiers présenté dans la [Section 5.2](#), qui utilise lui aussi $5n$ qubits.

Oracle de préparation des états superposés:

Lorsque les n qubits sont superposés, 2^n états sont créés, cependant tous ces états ne correspondent pas forcément aux entiers de la liste. Ainsi, nous utiliserons l'opérateur de Grover comme décrit dans la [Section 4.7](#) pour amplifier les états correspondant aux éléments de la liste. Conformément à l'algorithme de Grover et comme démontré dans [25] le nombre d'itérations à effectué est de l'ordre de $\frac{\pi}{4} \sqrt{\frac{2^n}{N}} - \frac{1}{2}$ où 2^n correspond à l'ensemble dans lequel nous allons chercher les valeurs et N le nombre de valeurs à trouver et à amplifier.

L'Oracle de l'opérateur de Grover pour amplifier les états correspondants aux valeurs de la liste est une succession de portes Z multi-controlées permettant chacune de reconnaître un entier de la liste.

Voici par exemple l'Oracle pour la liste $L = [5, 1, 3]$:

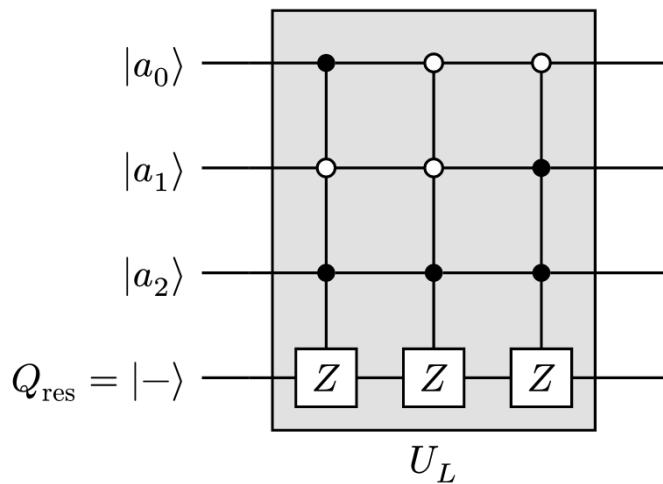


Figure 34: Circuit de l'Oracle pour la préparation des états avec $L = [5, 1, 3]$

On a donc bien $5 = a_0 a_1 a_2 = 101$ sur la première porte MCZ. Puis $1 = a_0 a_1 a_2 = 001$ pour la deuxième et enfin $3 = a_0 a_1 a_2 = 011$ pour la dernière. Ainsi, dès qu'un état correspondra à l'un des éléments de L , sa phase sera inversé par l'application d'une porte Z sur le qubit auxiliaire et sa probabilité sera amplifiée par l'opérateur de diffusion.

Oracle de comparaison des entiers:

Avant de créer le circuit complet de notre algorithme, nous allons revenir sur la description de l'Oracle pour la comparaison. Ainsi, la [Figure 35](#) montre à quoi ressemble le circuit de l'Oracle permettant de comparer 2 entiers sur n bits en appliquant une porte Z sur le qubit auxiliaire si $a < b$.

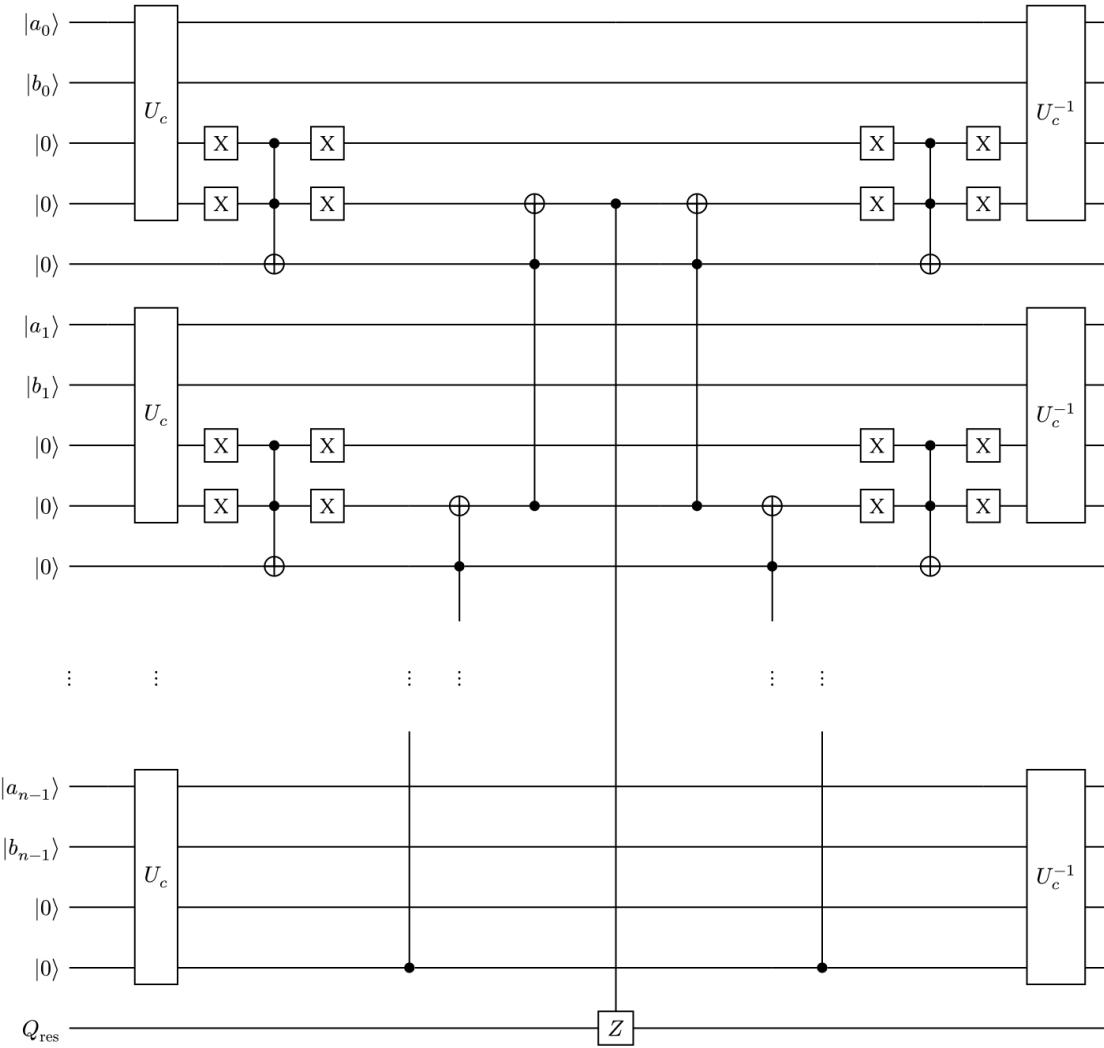


Figure 35: Circuit complet de l'Oracle de comparaison

Comme évoqué précédemment, l'Oracle doit appliquer une porte Pauli-Z sur le qubit Q_{res} du circuit si $a < b$, c'est à dire quand le deuxième qubit auxiliaire de la première porte U_c (du bit le plus fort dans l'écriture binaire) est à $|1\rangle$. On considérera que a est l'entier représenté par les états superposés et b est la valeur de référence y_i . On modifie alors la porte CNOT de la [Figure 31](#) sur le qubit Q_{res} par une porte CZ. On appliquera ensuite l'opérateur de diffusion pour amplifier les états qui vérifient la condition de comparaison.

Définition de l'opérateur de diffusion:

En reprenant les notations de la [Section 4.7](#), on note S_{ψ_H} l'opérateur de diffusion. Il est défini, pour un état $|\psi_H\rangle = H^{\otimes n}|0^n\rangle = |+^n\rangle$ par

$$\begin{aligned}
 S_{\psi_H} &= 2|\psi_H\rangle\langle\psi_H| - \hat{I} \\
 &= 2|\psi_H\rangle\langle\psi_H| - (\hat{H}\hat{I}\hat{H})^{\otimes n} \\
 &= 2H^{\otimes n}|0^n\rangle\langle 0^n|H^{\otimes n} - (\hat{H}\hat{I}\hat{H})^{\otimes n} \\
 &= H^{\otimes n}(2|0^n\rangle\langle 0^n| - \hat{I})H^{\otimes n}
 \end{aligned} \tag{77}$$

Voici un exemple de son implémentation dans un circuit avec une superposition de 4 qubits. Les qubits $|a_i\rangle$ ont subi au préalable des portes H pour les superposer, puis l'Oracle pour marquer ceux qui doivent l'être et qui doivent être amplifiés dans cet opérateur:

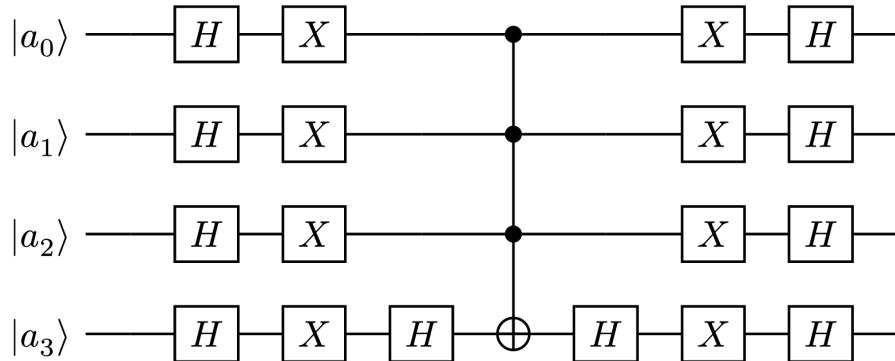


Figure 36: Circuit de l'opérateur de diffusion pour $n = 4$

Cet opérateur de diffusion est celui de Grover, il sera à la fois utilisé pour amplifier les états après le passage de l'Oracle de préparation des états et aussi par l'Oracle de comparaison des entiers.

Circuit complet:

Le circuit présenté dans cette section est le circuit utilisé à chaque itération de l'algorithme de recherche du minimum. On note \hat{G} l'opérateur qui sera répété $g \simeq \frac{\pi}{4} \sqrt{\frac{2^n}{N}} - \frac{1}{2}$ fois pour trouver les N états parmi les 2^n états créés par la superposition et les amplifier avec S_{ψ_H} l'opérateur de diffusion. On note également \hat{P} l'opérateur qui sera répété $p \simeq \frac{\pi}{4} \sqrt{\frac{2^n}{N}} - \frac{1}{2}$ fois, qui contient l'oracle de comparaison des entiers présenté sur la [Figure 35](#) et S_{ψ_H} l'opérateur de diffusion.

On retrouve $|y_i\rangle$ qui est la valeur de référence de la comparaison lors de l'Oracle de \hat{P} , initialisé comme décrit dans la [Section 5.2.1](#), qui jouera le rôle de b dans ce dernier. Ainsi, \hat{G} appliquera une porte Pauli-Z sur le qubit auxiliaire $|-\rangle$ lorsque a sera un élément de la liste L , et \hat{P} fera de même lorsque que a sera inférieur à b c'est à dire que l'état comparé a , est plus petit que l'état de référence $b = |y_i\rangle$.

À noter que $|y_i\rangle$ est codé sur n qubits, et que, du fait de sa construction, les états créés par superposition sont eux aussi des états sur n qubits. De plus, on ne représente pas les qubits auxiliaires de l'Oracle dans les entrées du circuit pour gagner en clareté.

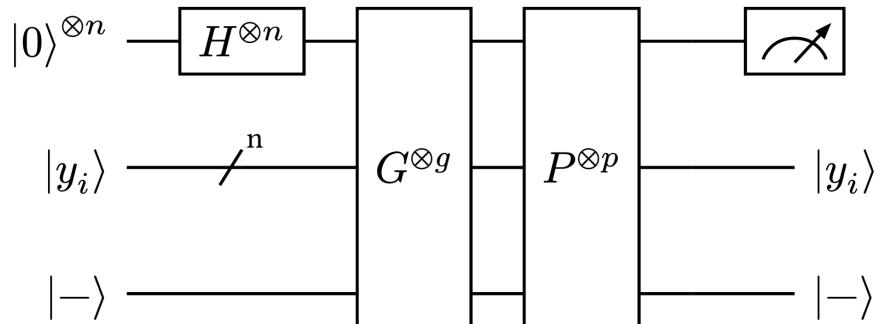


Figure 37: Circuit complet pour le marquage des éléments de L inférieurs à y_i

Avec l'opérateur \hat{G} qui s'applique sur les 2^n états superposés et qui applique une porte Z sur le qubit auxiliaire pour marquer les états présents dans la liste. Après la diffusion, le qubit auxiliaire est décalculé pour le prochain opérateur \hat{P} . À noté que $U_L = U_L^{-1}$.

Et l'opérateur \hat{P} qui s'applique sur tous les qubits du circuit et qui, à l'aide de qubits auxiliaires internes, permet d'appliquer une porte Z sur le qubit auxiliaire quand la condition de comparaison est validée. L'oracle de comparaison est explicité dans la [Figure 35](#). Voici un détail de chacun des deux opérateurs \hat{G} et \hat{P} .

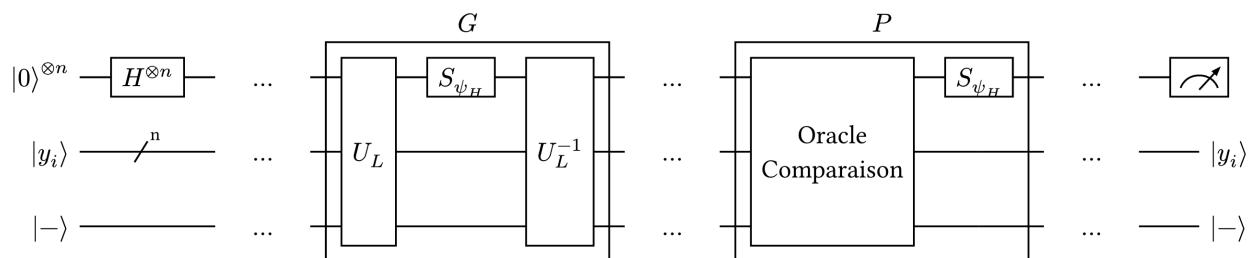


Figure 38: Détail des opérateurs \hat{G} et \hat{P}

Ce circuit permet donc de marquer les entiers inférieurs à une certaine valeur y_i . Il y a n qubits pour la superposition des entiers de la liste, n autres qubits pour la valeur de référence y_i et $3n$ qubits auxiliaires dont $3n - 1$ pour l'Oracle de comparaison des entiers et 1 pour inverser le signe des états marqués par les différents Oracles. Les mesures permettent de récupérer des n -qubits, dont les états les plus probables sont les entiers inférieurs à y_i .

Comme les opérateurs \hat{G} et \hat{P} sont répétés environ $\frac{\pi}{4}\sqrt{\frac{2^n}{N}} - \frac{1}{2}$ fois, on peut estimer la complexité du circuit par $C(n, N) = \frac{\pi}{4}\sqrt{\frac{2^n}{N}} - \frac{1}{2}$, dont on peut représenter la densité en fonction de n et N :

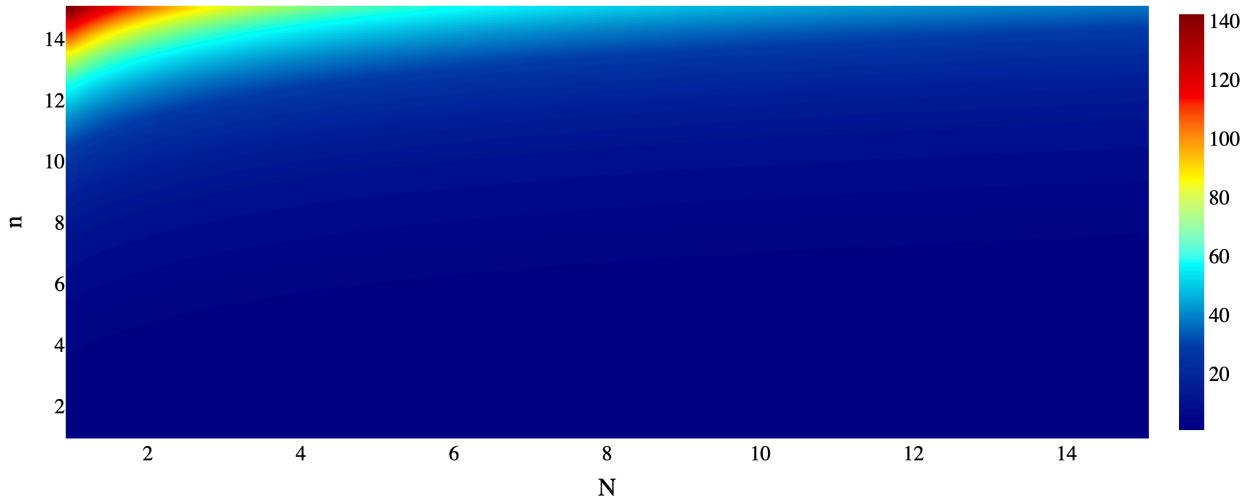


Figure 39: Complexité du circuit en fonction de n et N

La variation de couleurs indique l'augmentation de la complexité : les zones avec des couleurs plus froides (vers le bleu) indiquent une complexité plus faible, tandis que les zones avec des couleurs plus chaudes (vers le rouge) indiquent une complexité plus élevée. Cette représentation visuelle permet d'identifier rapidement comment la complexité évolue en fonction de la taille de la liste N et du nombre de bits n . À partir de ce graphique, on observe que :

- Pour des valeurs faibles de N et des valeurs élevées de n , la complexité augmente rapidement, ce qui est indiqué par les zones plus chaudes. Cela reflète la dominance de la complexité 2^n dans ces régions.
- À mesure que N augmente, la complexité diminue, surtout pour des valeurs plus faibles de n .
- On remarque que plus la liste est grande, plus la complexité est faible.

Spécification de l'algorithme:

Étant donné que cet algorithme de recherche du minimum dans une liste d'entiers codés sur n bits nécessite $5n$ qubits, alors dans un ordinateur quantique de k qubits on pourrait avoir une liste avec des entiers entre 0 et $2^k - 1$.

Test du circuit sur une itération:

Nous allons ici présenter les résultats des différents tests effectués avec les opérateurs \hat{G} et \hat{P} explicités dans la Figure 38. L'ensemble des circuits de ces tests sont réalisés sur le simulateur sans bruit **simulator_mps** de la plateforme IBM Cloud.

Pour l'exemple, voici une liste L :

$$L = [12, 7, 3, 18] \quad (78)$$

C'est une liste de taille 4 dont le nombre de bits nécessaires pour représenter ses entiers est 5. Après préparation des états, nous avons donc $2^5 = 32$ états superposés représentants les valeurs de 0 à 31 (intervalle qui contient nos entiers). La valeur de y_i est choisi arbitrairement à 12 (y_i est toujours une valeur appartenant à la liste), et le qubit auxiliaire Q_{res} est initialisé à $|-\rangle$.

Si nous lançons $g = 1$ fois l'opérateur \hat{G} qui filtre les entiers de la liste (sans l'opérateur \hat{P}) nous obtenons les probabilités suivantes:

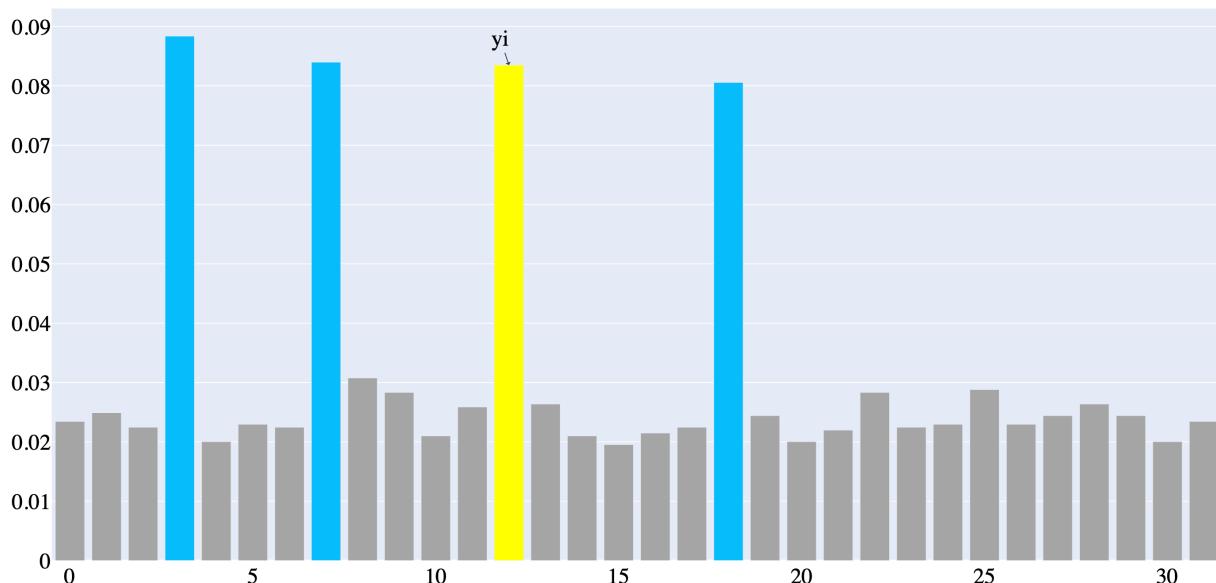


Figure 40: Probabilités des états après l'opérateur \hat{G}

Les barres en bleu et jaune correspondent aux entiers de la liste L et ils sont bien amplifiés comme cela est prévu par l'opérateur \hat{G} explicité dans la Figure 34.

Maintenant, nous allons tester avec $p = 1$ fois l'opérateur \hat{P} (sans l'opérateur \hat{G}) pour voir le résultat. Pour rappel, $y_i = 12$. Voici le résultat:

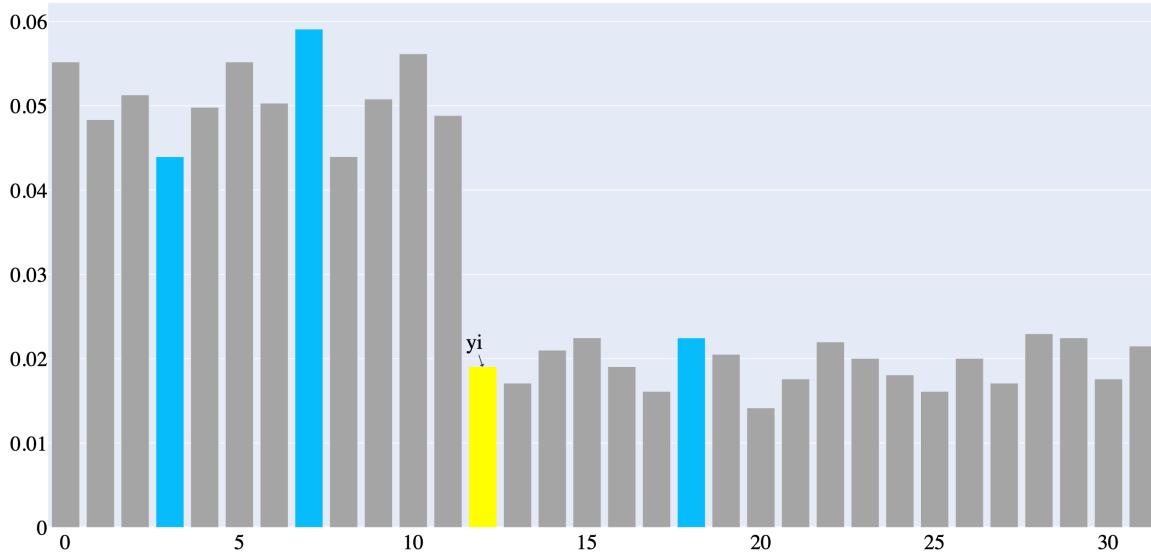


Figure 41: Probabilités des états après l'opérateur \hat{P}

On remarque que les états étant inférieurs strictement à $y_i = 12$ sont amplifiés, ce qui est bien le résultat attendu de l'Oracle décrit dans la [Figure 35](#).

À présent, nous allons combiner les opérateurs \hat{G} et \hat{P} pour voir le résultat sur la liste L :

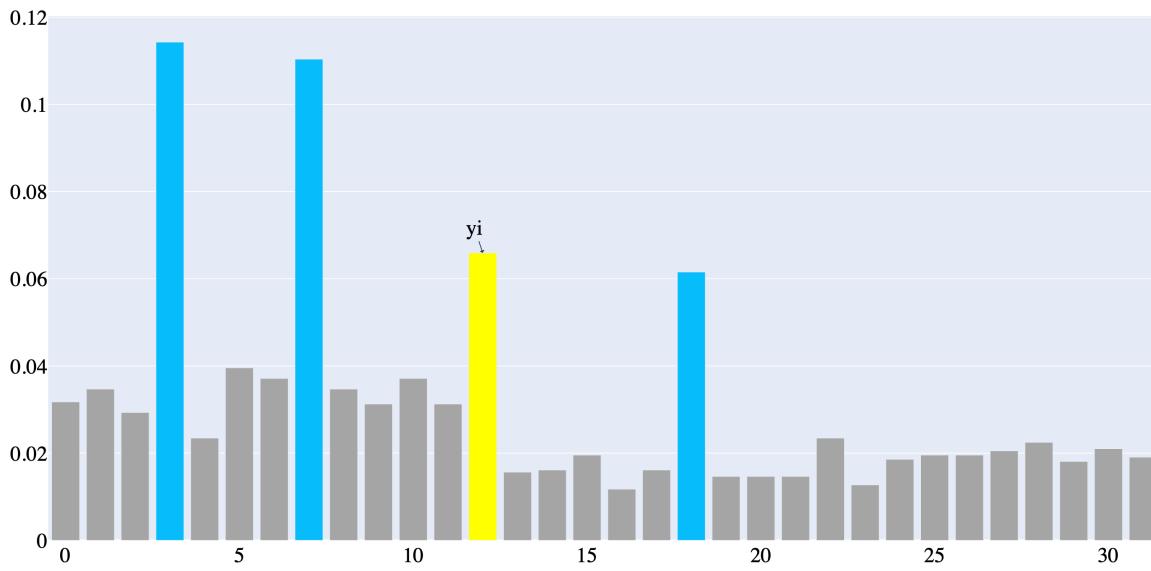


Figure 42: Probabilités des états après la combinaison des opérateurs \hat{G} et \hat{P}

L'utilisation combinée de \hat{G} et \hat{P} a amplifié deux fois plus les états qui à la fois sont présents dans la liste et aussi les états inférieurs à $y_i = 12$. On retrouve alors les valeurs $|3\rangle$ et $|7\rangle$. La nouvelle valeur y_{i+1} devient alors l'état le plus probable, qui est ici $|3\rangle$.

Test de l'algorithme complet:

L'algorithme complet consiste donc à choisir une première valeur aléatoire dans la liste, notée y_0 , et de donner au circuit présenté dans la Figure 37 la liste L ainsi que cet élément y_0 . On relance le circuit t fois avec à chaque fois la valeur y_i précédente. L'algorithme s'arrête quand le minimum trouvé y_i n'est plus un élément de la liste ou alors que nous avons atteint le maximum d'itération fixé à \sqrt{N} , auquel cas nous choisissons comme minimum la dernière valeur calculée. L'ensemble des circuits de ces tests sont également réalisés sur le simulateur sans bruit **simulator_mps** de la plateforme IBM Cloud.

En reprenant la liste précédente L tel que:

$$L = [3, 4, 7, 8, 10, 13, 5, 12, 1, 11, 6, 2, 9] \quad (79)$$

Et en choisissant une valeur aléatoire $y_0 = 4$, un nombre d'itération de \hat{G} égale à 1 et un nombre d'itération de \hat{P} égale à 1, voici la sortie du circuit en terme de probabilité:

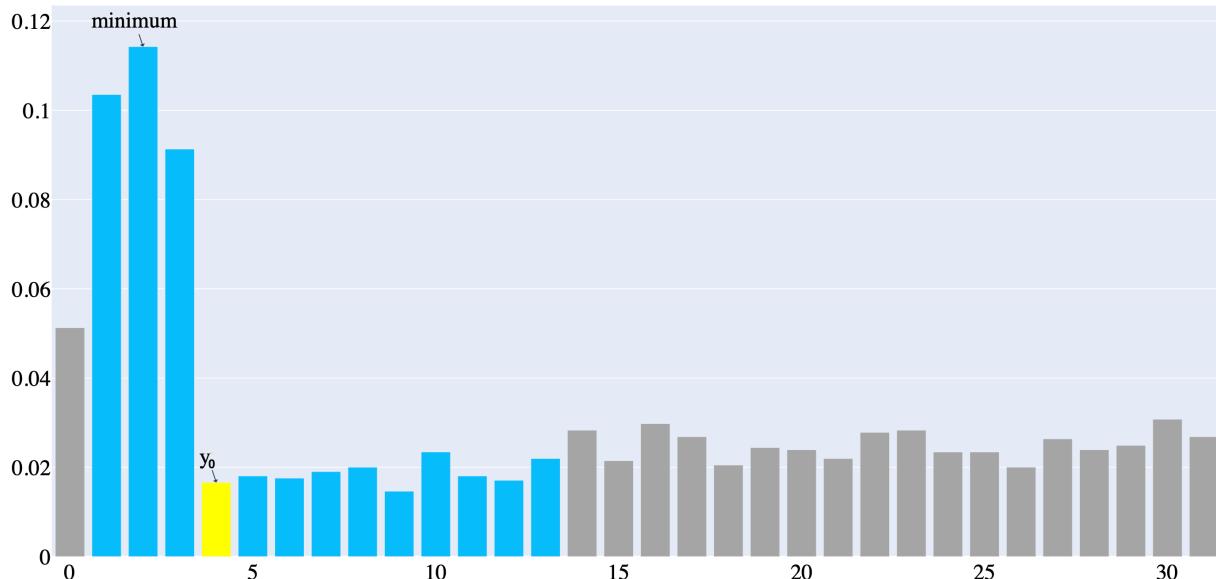


Figure 43: Première itération de l'algorithme de recherche du minimum

Après cette première itération, les états les plus probables sont donc ceux qui sont dans L et qui sont inférieurs strictement à $y_0 = 4$. Pour la prochaine itération on choisit ainsi $y_1 = 2$.

Voici les résultats de la deuxième itération:

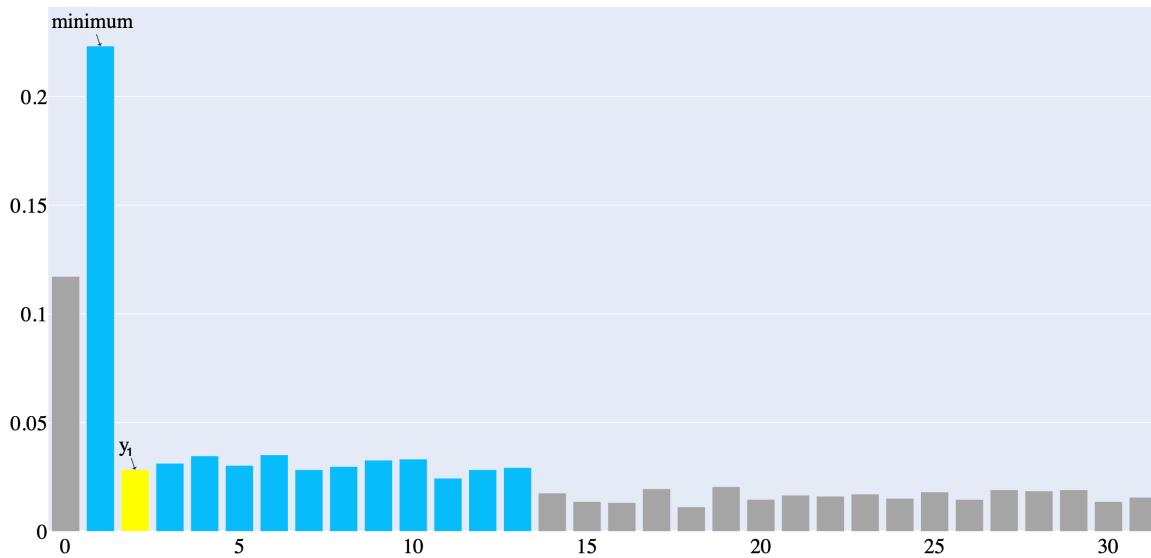


Figure 44: Deuxième itération avec $y_1 = 2$

Après cette deuxième itération, les états les plus probables sont restreints à $|1\rangle$. On va donc choisir $y_2 = 1$ comme nouvelle valeur. Voici les résultats de la troisième itération:

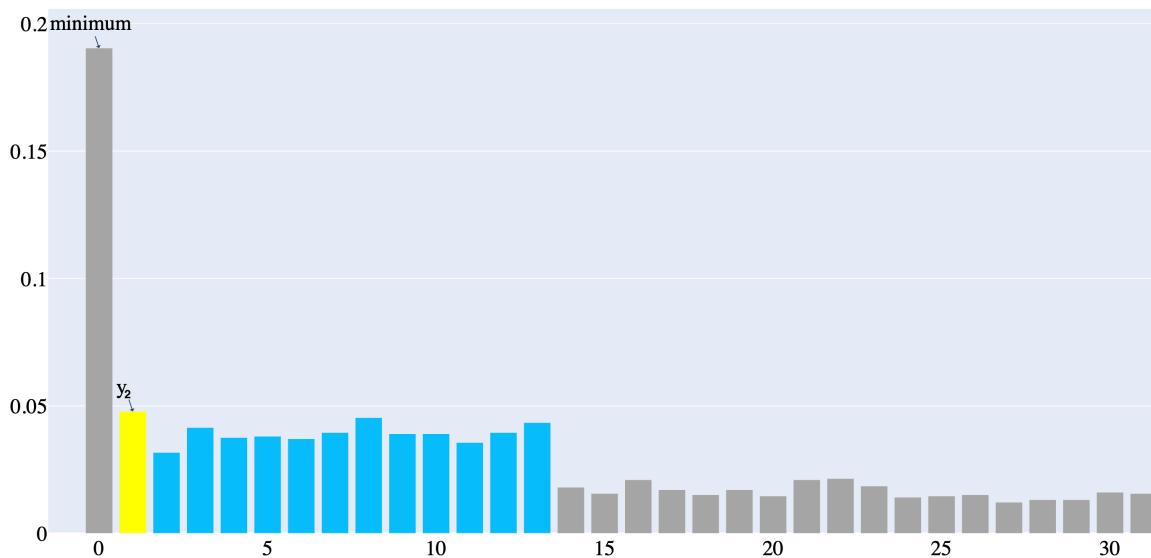


Figure 45: Deuxième itération avec $y_2 = 1$

À ce stade, l'état le plus probable correspond à $|0\rangle$, mais 0 n'étant pas dans \mathcal{L} on va donc conclure en disant que le minimum de la liste est la dernière valeur trouvée, et ainsi la valeur 1 est le résultat de l'algorithme. Dans cet exemple 1 est le véritable minimum.

6.. ALGORITHME Q-MEANS

Nous allons à présent assembler les sous-routines quantiques développées dans la section précédente pour construire l'algorithme QMeans. Cette version est donc une version hybride, qui alterne entre calculs classiques et calculs quantiques.

6.1.. Initialisation Q-Means++

L'algorithme Q-Means vient avec son initialisation dérivée du kmeans++ expliquée dans la [Section 3](#), nommée Q-Means++. Le principe est le même que son homologue classique, mis à part le calcul des distances qui se fait de manière quantique.

On commence par choisir un point aléatoire parmi les données, qui sera le premier centroïde du modèle. Ensuite, pour chaque nouveau centroïde à sélectionner, le processus est un peu plus complexe et utilise les capacités de calcul quantique. Plus précisément, pour chaque point de données, on calcule la distance entre ce point et l'ensemble actuel de centroïdes à l'aide du circuit détaillé dans la [Section 5.1](#). Et on calcule le minimum de cette liste.

On obtient ainsi l'ensemble des distances minimales de chaque point suivant:

$$\text{distances} = [d(x_1, C), d(x_2, C), \dots, d(x_n, C)] \quad (80)$$

où $d(x, C)$ est la distance du point x au centroïde le plus proche C .

Après avoir obtenu ces distances minimales pour tous les points, on les normalise de manière classique pour obtenir une distribution de probabilités. Cette distribution est utilisée pour sélectionner le prochain centroïde de manière probabiliste, de façon à favoriser les points qui sont plus éloignés des centroïdes déjà sélectionnés. On calcule donc la probabilité pour qu'un point x_i soit sélectionné en tant que cluster.

$$p(x_i) = \frac{d(x_i, C)}{\sum_{j=1}^n d(x_j, C)} \quad (81)$$

En répétant cette procédure jusqu'à ce que le nombre requis de centroïdes soit atteint, on finalise l'initialisation des centroïdes pour l'algorithme Q-Means.

6.2.. Estimation de la distance des centroïdes

À chaque itération on estime la distance entre chaque point et les centroïdes des clusters. Pour cela nous utiliserons la procédure quantique décrite dans la [Section 5.1](#). Elle permet, à partir d'un point X et de n centroïdes C_i , d'estimer la distance entre X et chacun des C_i . Pour chacun des points X des données, nous lui attribuons un cluster, suivant la section suivante.

6.3.. Affectation du cluster

Pour chaque point X des données nous obtenons donc une estimation de la distance avec tous les centroïdes. Ne reste plus qu'à calculer la distance minimale pour connaître le centroïde le plus proche et ainsi affecter au point le bon cluster. Pour le calcul du minimum nous utilisons la procédure détaillée dans la [Section 5.3](#).

7.. PSEUDO-CODE Q-MEANS

Initialisation Q-Means++:

On initialise les centroïdes suivant l'algorithme QMeans++ décrit dans la [Section 6.1](#).

Pour n clusters, il faudra $3n$ qubits pour l'estimation des distances. Pour une liste d'entiers sur n bits, il faudra $5n$ qubits pour trouver le minimum.

Tant que les centroïdes continuent à changer de position:

Étape 1: Estimation des distances aux centroïdes (quantique)

Pour estimer les distances de manière quantique, entre un point x et un centroïde y on utilise le produit scalaire comme indicateur de distance, comme expliqué dans la [Section 5.1.1](#)

On calcule ainsi:

$$P(1_{\text{aux}}) = \frac{1}{2} - \frac{1}{2}\langle x, y \rangle^2 \quad (82)$$

On obtient donc pour chaque point, une estimation de la distance avec tous les centroïdes.

Pour n clusters, il faudra $3n$ qubits.

Étape 2: Affectation du cluster (quantique)

On va calculer, pour chaque point, le minimum des distances avec les centroïdes pour les associer au bon cluster à l'aide de l'algorithme Quantum Bit String Comparator associé à l'algorithme de **Dürr-Hoyer [23]**, comme expliqué dans la [Section 5.3](#).

Pour une liste d'entiers sur n bits, il faudra $5n$ qubits.

Étape 3: Mise à jour des centroïdes (classique)

Chaque nouveau centroïde est le barycentre de l'ensemble de points du cluster correspondant. Ainsi :

$$c_i = \frac{1}{\#C_i} \sum_{x \in C_i} x \quad (83)$$

Avec $\#C_i$ le nombre de points du cluster C_i .

Résultat:

On obtient au final la position des centroïdes.

8.. DISCUSSION

8.1.. Restriction sur le nombre de qubits

L'un des défis majeurs de l'implémentation des sous-routines du Q-Means concerne les restrictions imposées par le nombre de qubits disponibles ainsi que l'utilisation de la QRAM. En effet, l'ordre de grandeur du nombre de qubits est environ une centaine. C'est pour cela notamment que le calcul du minimum d'une liste d'entiers a nécessité la création d'un circuit reposant sur des Oracles permettant de traiter énormément d'entiers avec peu de qubits. Pour la QRAM, n'étant encore que théorique, nous avons dû contourner son utilisation en itérant un circuit et en mettant à jour les entrées au fur et à mesure.

Nous pouvons ainsi calculer le minimum d'une liste de taille quelconque, avec des entiers entre 0 et $2^{20} - 1 = 1.048.575$ et calculer la distance entre une donnée et 33 clusters en n'utilisant que 100 qubits par exemple.

8.2.. Nombre d'itérations de l'opérateur \hat{P}

Dans la [Section 5.3](#) nous avons fixé le nombre d'itérations de l'opérateur \hat{P} à $p = \frac{\pi}{4} \sqrt{\frac{2^n}{N}} - \frac{1}{2}$. Cependant, au fur et à mesure des itérations, le nombre de valeurs à marquer se réduit et donc on pourrait penser que ce nombre est divisé par 2 à chaque itération. C'est à dire qu'on aurait $N \times 2^{-i}$ valeurs à amplifier avec i de 0 à \sqrt{N} le nombre d'itérations. Et ainsi, le nombre d'itérations p de l'opérateur \hat{P} serait donc à la i -ème itération:

$$p = \frac{\pi}{4} \sqrt{\frac{2^{n+i}}{N}} - \frac{1}{2} \quad (84)$$

Des tests et simulations supplémentaires seraient nécessaires pour confirmer cette théorie.

8.3.. Implémentation

Cet article est lié à un [dépôt Github](#) dans lequel a été codé Kmeans et δ -kmeans à des fins de comparaison, mais aussi QMeans avec toutes les sous-routines présentées dans les sections précédentes. Cependant, la simulation d'un tel algorithme n'est pas possible avec un grand nombre de qubits car cela nécessite une puissance de calcul trop importante. Et l'utilisation d'ordinateurs quantiques réels est contrainte par le temps d'attente pour accéder aux ordinateurs quantiques sur la plateforme d'IBM. Malgré cela, chaque sous-routine quantique est utilisable indépendamment avec un nombre raisonnable de qubits.

8.4.. État de l'art de l'algorithme

L'implémentation du Q-Means exposée dans cet article se base sur une approche naïve, recourant à l'utilisation séquentielle de circuits quantiques. Cette méthode contraste avec la théorie avancée par Kerenidis et al. [\[1\]](#), qui privilégie l'emploi d'un unique circuit quantique. En conséquence, bien que notre algorithme soit conforme aux critères d'efficacité attendus, il ne parvient pas à satisfaire les exigences en matière de rapidité de calcul.

9.. CONCLUSION

En synthèse, cet article a exploré les possibilités offertes par l'algorithme Q-Means, une variante quantique de l'approche classique de clustering K-means. Émergeant de l'intersection de l'informatique quantique et du machine learning.

Ce travail diverge de l'approche théorique initiale proposée par Kerenidis et al. [1], en ce que nous utilisons une méthode séquentielle plutôt qu'un unique circuit quantique. Bien que cela permette une implémentation plus simple et conforme aux contraintes actuelles des ordinateurs quantiques, elle ne réalise pas le plein potentiel de rapidité que la théorie suggère.

Pour l'anecdote, ce projet a nécessité le lancement de pas moins de 100.000 circuits quantiques sur les serveurs d'IBM.

REFERENCES

- [1] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning", vol. 32, p., 2019, [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/16026d60ff9b54410b3435b403af226-Paper.pdf
- [2] D. Arthur and S. Vassilvitskii, "K-Means++: The Advantages of Careful Seeding", p. 1027, 2007, [Online]. Available: <https://dl.acm.org/doi/10.5555/1283383.1283494>
- [3] R. Hathaway, J. Bezdek, and J. Huband, "Maximin Initialization for Cluster Analysis", 2006, pp. 14–26. doi: [10.1007/11892755_2](https://doi.org/10.1007/11892755_2).
- [4] Caltech, "What Is Superposition and Why Is It Important?". [Online]. Available: <https://scienceexchange.caltech.edu/topics/quantum-science-explained/quantum-superposition>
- [5] Caltech, "What Is Entanglement and Why Is It Important?". [Online]. Available: <https://scienceexchange.caltech.edu/topics/quantum-science-explained/entanglement>
- [6] O. Ezratty, "Understanding Quantum Technologies 2023", 2023. [Online]. Available: <https://www.oezratty.net/wordpress/2023/understanding-quantum-technologies-2023/>
- [7] L. K. Grover, "A fast quantum mechanical algorithm for database search". [Online]. Available: <https://arxiv.org/abs/quant-ph/9605043>
- [8] E. Bourreau, G. Fleury, and P. Lacomme, *Introduction à l'informatique quantique: Apprendre à calculer sur des ordinateurs quantiques avec Python*, 1ère édition. Eyrolles, 2022, p. 45. [Online]. Available: <https://www.editions-eyrolles.com/Livre/9782416006531/introduction-a-l-informatique-quantique>
- [9] G. Fleury and P. Lacomme, *Les algorithmes de base de l'informatique quantique - Tome 2*, 2eme édition. Eyrolles, 2023. [Online]. Available: <https://www.eyrolles.com/Informatique/Livre/les-algorithmes-de-base-de-l-informatique-quantique-tome-2-9782416011726/>
- [10] IBM, "IBM Quantum Composer". [Online]. Available: <https://quantum-computing.ibm.com/composer>
- [11] IonQ, "How Do Trapped Ion Computers Work?", 2023, [Online]. Available: <https://ionq.com/resources/how-do-trapped-ion-computers-work#:~:text=How%20Do%20Trapped%20Ion%20Computers,the%20boundaries%20of%20what%27s%20possible>
- [12] U. de Reims Champagne-Ardenne, "ROMEO HPC Center". [Online]. Available: <https://romeo.univ-reims.fr/pages>
- [13] A. Ballon, "Trapped ion quantum computers", *PennyLane Demos*, 2021, [Online]. Available: https://pennylane.ai/qml/demos/tutorial_trapped_ions
- [14] A. Ballon, "Quantum computing with superconducting qubits". [Online]. Available: https://pennylane.ai/qml/demos/tutorial_sc_qubits
- [15] D. M. Ginsberg, "superconductivity". [Online]. Available: <https://www.britannica.com/science/superconductivity>
- [16] A. F. Kockum and F. Nori, "Quantum Bits with Josephson Junctions", in *Fundamentals and Frontiers of the Josephson Effect*, F. Tafuri, Ed., Cham: Springer International Publishing, 2019, pp. 703–741. doi: [10.1007/978-3-030-20726-7_17](https://doi.org/10.1007/978-3-030-20726-7_17).
- [17] R. Tahir-Kheli, "Cooper Pair", in *General and Statistical Thermodynamics*, Cham: Springer International Publishing, 2020, pp. 507–517. doi: [10.1007/978-3-030-20700-7_13](https://doi.org/10.1007/978-3-030-20700-7_13).
- [18] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum Random Access Memory", *Physical Review Letters*, vol. 100, no. 16, Apr. 2008, doi: [10.1103/physrevlett.100.160501](https://doi.org/10.1103/physrevlett.100.160501).
- [19] S. Xu, C. T. Hann, B. Foxman, S. M. Girvin, and Y. Ding, "Systems Architecture for Quantum Random Access Memory", in *56th Annual IEEE/ACM International Symposium on Microarchitecture*, in *MICRO '23*. ACM, Oct. 2023. doi: [10.1145/3613424.3614270](https://doi.org/10.1145/3613424.3614270).

- [20] S. DiAdamo, C. O'Meara, G. Cortiana, and J. Bernabe-Moreno, "Practical Quantum K-Means Clustering: Performance Analysis and Applications in Energy Grid Classification", *IEEE Transactions on Quantum Engineering*, vol. 3, pp. 1–16, 2022, doi: [10.1109/tqe.2022.3185505](https://doi.org/10.1109/tqe.2022.3185505).
- [21] "Valeur moyenne (quantique)". [Online]. Available: [https://fr.wikipedia.org/wiki/Valeur_moyenne_\(quantique\)](https://fr.wikipedia.org/wiki/Valeur_moyenne_(quantique))
- [22] D. Oliveira and R. Ramos, "Quantum bit string comparator: Circuits and applications", *Quantum Computers and Computing*, vol. 7, p., 2007, [Online]. Available: https://www.researchgate.net/publication/228574906_Quantum_bit_string_comparator_Circuits_and_applications
- [23] C. Durr and P. Hoyer, "A Quantum Algorithm for Finding the Minimum". [Online]. Available: <https://arxiv.org/abs/quant-ph/9607014>
- [24] A. S. Albino, L. Q. Galvão, E. Hansen, M. Q. N. Neto, and C. Cruz, "Quantum algorithm for finding minimum values in a Quantum Random Access Memory". [Online]. Available: <https://arxiv.org/abs/2301.05122>
- [25] E. Borbely, "Grover search algorithm". [Online]. Available: <https://arxiv.org/abs/0705.4171>