PROJET SCIENTIFIQUE COLLECTIF
RAPPORT FINAL

Informatique quantique
&
Problème du voyageur de commerce

MAP10

Avril 2021

Tuteur : Georges Uzbelger - IBM
Coordinateur : Teddy Pichard
Nathanaël Cuvelle-Magar, Sébastien Draux, Théo Ferry,
Xiyao Li, Hugo Massonnat

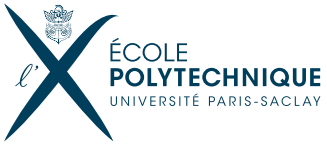


Table des matières

1 Introduction

Ce rapport, qui a pour but de dresser le **bilan** de notre PSC, poursuit un double objectif :

— présenter les **résultats obtenus**, en dressant un portrait des acquis et des réalisations ;

— expliciter la **démarche** mise en œuvre au cours du projet, tant au niveau **scientifique** que sur le plan du **fonctionnement de groupe**.

La présentation des acquis et des réalisations, cœur de ce document, sera ainsi articulée de sorte à mettre en évidence la progression logique entre les étapes du projet. La démarche scientifique pourra par conséquent se lire en creux de cette progression, le fonctionnement de groupe étant quant à lui traité dans une partie consacrée en fin de rapport.
Dans un premier temps, nous commencerons donc par poser les bases de notre étude en définissant rigoureusement le **problème du voyageur de commerce** (TSP, pour **Traveling Salesman Problem**).
Cet énoncé nous conduira ensuite à présenter rapidement différentes familles d'**algorithmes de résolution classiques** mettant en lumière la difficulté centrale du problème, c'est-à-dire la recherche d'un optimum dans un espace de taille $n!$. Les **approches heuristiques**, dont plusieurs se prêtent de façon naturelle à une **transposition quantique** (par exemple l'exploration probabiliste), nous permettront alors d'illustrer la pertinence d'une approche via l'informatique quantique, dont la partie suivante présentera les briques de base en termes d'états et de portes logiques.
Un résumé des principales caractéristiques de cette nouvelle informatique, au travers d'une présentation succincte d'**algorithmes quantiques canoniques**, achèvera enfin de justifier l'adéquation d'une telle démarche au problème du voyageur de commerce.
Nous présenterons ensuite le fonctionnement de l'algorithme que nous avons choisi d'utiliser, le **QAOA** (**Quantum Approximate Optimization Algorithm**), dont l'implémentation fera l'objet des deux parties suivantes. La première traitera d'une **approche naïve** puis, d'une **version optimisée**, en s'intéressant dans les deux cas à l'**interprétation des résultats** et au **cadre théorique** sous-jacent. Ces interprétations nous conduiront finalement à proposer une **dernière implémentation**, détaillée dans une seconde partie, et qui constitue la version la plus aboutie du code de ce projet. Tous les algorithmes développés sont disponibles sur github (lien github) et ont été implémentés sur **Qiskit**, le kit de développement d'IBM dédié à l'informatique quantique (lien Qiskit).
Après une réflexion sur notre **fonctionnement de groupe** au travers de cette année de PSC, nous conclurons finalement ce rapport par un résumé de notre démarche en termes d'implémentation et des **améliorations possibles de ce projet**.

2 Présentation du problème et approche quantique

2.1 Le TSP : définition, complexité et applications

On peut définir le problème du voyageur de commerce comme suit : étant donné un ensemble de n villes et les distances entre celles-ci, trouver le chemin de longueur minimale passant une et une seule fois par chaque ville et revenant à son point de départ.
Les paragraphes suivants permettront de préciser la définition, la complexité et les applications de ce problème.

2.1.1 Définition

Formellement, on peut définir le TSP de la façon suivante en s'appuyant sur le cours INF412 de l'école [Bou] :

Definition 1. *Soit n un entier et W une matrice d'entiers de taille $n \times n$. Déterminer la permutation π de $\{0, 1, ..., n-1\}$ qui minimise $\sum_{0 \leq i \leq n-1} W_{\pi(i)\pi(i+1)}$ (où on confond les indices 0 et n).*

La version décisionnelle du TSP présente également un intérêt algorithmique en tant que problème NP-complet (ce que nous développons dans la partie suivante) :

Definition 2. *Soit n un entier, W une matrice d'entiers de taille $n \times n$, et k un entier. Déterminer s'il existe permutation π de $\{0, 1, ..., n-1\}$ qui vérifie $\sum_{0 \leq i \leq n-1} W_{\pi(i)\pi(i+1)} \leq k$.*

Si on considère que W est la matrice de poids d'un graphe pondéré, il s'agit de trouver le chemin de longueur minimale qui visite une et une seule fois chaque sommet en revenant à son point de départ, c'est-à-dire le cycle hamiltonien de longueur minimal. L'indice $\pi(i)$ est le sommet visité à l'étape i du chemin.

2.1.2 Complexité

Definition 3. *NP, ou Non-Deterministic Polynomial, est la classe des problèmes décidables en temps polynomial avec une machine de Turing non-déterministe. Cela correspond également aux problèmes qui possèdent un vérificateur en temps polynomial.*

On dit qu'un problème A est plus difficile qu'un problème B s'il existe une fonction calculable (appelée réduction) qui transforme toute instance de B en une instance de A et telle que la solution pour le problème B se déduise de la solution pour A. Pour plus de détails, on pourra se référer à [Bou].

Definition 4. *Un problème A est dit :*

- *NP-difficile s'il est plus difficile que tout autre problème de NP ;*
- *NP-complet s'il est NP-difficile et qu'il appartient à NP.*

Le problème du voyageur de commerce est un **problème NP-difficile**. En effet, étant donné une permutation, il n'est pas facile de vérifier qu'elle réalise le minimum (en temps polynomial). Sa version décisionnelle est quant à elle **NP-complet**. Il est en effet facile, étant donné une solution, de vérifier si elle est correcte ; il suffit de vérifier que le cycle trouvé a une longueur inférieure à la borne choisie. Elle est de plus NP-complet, c'est-à-dire que tout problème NP peut se ramener à une instance de ce problème en temps polynomial (la preuve est donnée dans [Bou]).
Il existe de nombreuses variantes du TSP. Par exemple, et ce sera le cas pour notre étude, on peut considérer une matrice de distance W est symétrique. On peut aussi définir le problèmes avec des poids réels quelconques (et non nécessairement entiers) ou encore des poids binaires. En réalité, sauf cas très particulier, toutes les variantes du TSP sont algorithmiquement équivalentes.

La résolution de ce problème est donc très coûteuse : dans le pire des cas, la complexité temporelle de l'algorithme naïf (tester tous les chemins) varie factoriellement avec le nombre de sommets. Il n'est *a priori* pas évident qu'il existe une solution exponentielle au problème.

2.1.3 Applications

La résolution du TSP a plusieurs applications, telles que la **planification**, la **logistique**, la **fabrication de puces** et le **séquençage de l'ADN**. Dans ces applications, le concept de «sommet» est utilisé pour désigner les clients, les soudures ou les fragments d'ADN, et le concept de «distance» fait référence au temps de trajet, au coût ou à une mesure de similitude entre les fragments d'ADN. Dans de nombreux scénarios d'applications (ressources limitées, temps limité, etc.), des contraintes supplémentaires peuvent être ajoutées.

2.2 Algorithmes classiques et intérêt d'une approche quantique

Dans un premier temps, nous avons principalement travaillé sur les algorithmes classiques (par opposition aux algorithmes quantiques) de résolution du TSP : méthodes exactes, algorithmes gloutons, algorithmes génétiques et algorithmes des fourmis. Nos études sur ces algorithmes nous ont permis d'avoir un premier aperçu de la complexité du TSP et de la difficulté importante représentée par l'optimisation des solutions existantes. Sans rentrer plus que nécessaire dans les détails techniques de ces algorithmes, on peut résumer leurs principes de fonctionnement de la façon suivante :

— Les **méthodes exactes** consistent en une exploration intelligente de l'espace des solutions. Ces méthodes permettent de trouver un optimum à chaque fois, mais malgré des essais d'optimisation, le temps de calcul augmente exponentiellement avec le nombre de sommets. Nous nous sommes notamment intéressés à l'algorithme de Held et Karp qui utilise le paradigme de la programmation dynamique.

— À l'inverse, les **algorithmes gloutons** sont une famille d'algorithmes donnant des approximations, qui peuvent s'avérer très éloignées de la solution optimale, mais qui ont une complexité moyenne plus acceptable. Ces algorithmes font les choix les plus avantageux selon leurs possibilités à un instant donné, sans se préoccuper des conséquences à

long terme résultant de ce choix. Par exemple dans le cas du TSP, si le voyageur choisit à chaque fois le sommet le plus proche de lui, on obtient une solution qui peut être satisfaisante, mais aussi assez éloignée de la solution optimale.

— Enfin, les **algorithmes génétiques** et les **algorithmes des fourmis** sont inspirés de la nature, respectivement de la théorie darwinienne de l'évolution et du comportement d'une colonie de fourmis. Ces deux types d'algorithmes d'optimisation commencent à partir d'un parcours non optimal (obtenu avec un algorithme glouton par exemple). Avant d'atteindre la condition d'arrêt, on fait boucler un processus de création et de sélection. Par exemple, lors de la recherche de nourriture, la liberté laissée aux fourmis permet une diversité de comportements les empêchant d'entrer dans une impasse et de faire des boucles sans fin, ce qui est une capacité d'innovation ; les retours positifs conservent de bonnes solutions et constituent une capacité d'apprentissage et de renforcement. La combinaison de la diversité et des retours positifs fait émerger des solutions optimales.

On peut se référer à [GP06] pour une étude complète du problème.
Le point important est que malgré cette diversité d'approches, il n'existe pas encore d'algorithme classique capable de résoudre le TSP en temps polynomial. Cela pousse ainsi à rechercher de nouvelles pistes de résolution, capables de franchir cette barrière de la complexité.
L'**approche quantique** en particulier apparaît être une option pertinente, en effet :

— elle semble pouvoir être **adaptée à ce problème**, comme en témoigne le caractère naturel de la transposition en informatique quantique de certains algorithmes classiques de résolution du TSP (notamment les marches aléatoires, via le phénomène de superposition d'états) ;

— il existe de nombreux exemples d'algorithmes quantiques ayant permis une **réduction drastique de complexité** par rapport aux approches classiques.

Ces deux raisons nous amènent ainsi à nous intéresser à des algorithmes quantiques avec l'espoir de diminuer la complexité du problème, notamment grâce au phénomène de superposition. La partie 4.1 donnera plusieurs exemples d'algorithmes quantiques canoniques, tels que les algorithmes de Shor et de Grover, ayant permis de telles améliorations de complexité.
Nous introduirons ensuite l'algorithme d'optimisation quantique que nous avons choisi d'utiliser pour résoudre le TSP (partie 4.2), avant de présenter en détail son fonctionnement (partie 5). Pour cela, nous utiliserons plusieurs notations et éléments de formalisme que la partie suivante va permettre d'introduire.

3 États et Portes quantiques : définitions et notations

En s'appuyant sur le cours PHY430 de l'Ecole [Jof], on peut donner la définition suivante d'un qubit.

Definition 5. *A l'instar d'un bit d'information qui peut prendre les deux valeurs 0 ou 1, un qubit est un système décrit par un espace de Hilbert sur le corps des complexes, de dimension deux, engendré par des états notés* $|0\rangle$ *et* $|1\rangle$.

Ce changement de paradigme se traduit notamment par la possibilité d'utiliser le parallélisme quantique pour créer des algorithmes dépassant les performances classiques. Les réalisations pratiques de tels systèmes peuvent être très variées : le spin d'un électron, le niveau d'énergie dans un atome ou encore la polarisation d'un photon en sont des exemples possibles. Pour ses ordinateurs quantiques, IBM a fait le choix de développer une technologie fondée sur les qubits supraconducteurs, plus particulièrement sur les transmons. Les qubits sont alimentés par des oscillateurs harmoniques utilisant des radio-fréquences allant de 5 à 10 GHz qui leur sont transmises par fils conducteurs électriques. Les opérations logiques sur ces qubits sont ainsi effectuées par le biais de jonctions Josephson contrôlées par un champ magnétique externe qui est une sorte de robinet pour manipuler la circulation du courant.

3.1 Système d'un seul qubit

3.1.1 Bra, ket et sphère de Bloch

Le but de cette sous-partie est de définir précisément ce qu'est l'état quantique d'un qubit et d'introduire le formalisme algébrique pour manipuler mathématiquement ces états. Pour cela, on commence par définir les deux états de base $|0\rangle$ et $|1\rangle$:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Un état quantique est une combinaison linéaire normée de ces deux états : $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ où $\alpha, \beta \in \mathbb{C}$ et $|\alpha|^2 + |\beta|^2 = 1$. L'espace des états d'un qubit est un espace de Hilbert de dimension 2 sur le corps des complexes. L'état $|\psi\rangle$ se lit "*ket* ψ".
On note $\langle\psi| =^T \overline{|\psi\rangle}$, ce qui se lit "*bra* ψ". Cette notation est particulièrement commode pour calculer des produits scalaires : le produit scalaire entre deux états quantiques $|\psi\rangle$ et $|\phi\rangle$ se note alors $\langle\psi|\phi\rangle$.
Les états quantiques différant seulement d'un facteur de phase sont indiscernables l'un de l'autre : aucune expérience physique ne permet de distinguer l'état $|\psi\rangle$ de l'état $|\psi'\rangle = e^{ia}\,|\psi\rangle$, quel que soit $a \in \mathbb{R}$. Ainsi, on confondra ces états. On peut donc se restreindre aux états $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ avec $\alpha \in \mathbb{R}_+$. On peut alors écrire $\alpha = \cos(\theta/2)$ pour un certain $\theta \in [0, \pi]$. Alors $|\beta| = \sin(\theta/2)$ d'où $\beta = \sin(\theta/2)e^{i\varphi}$ pour un certain $\varphi \in [0, 2\pi]$. On a donc montré que tout état quantique s'écrivait à un facteur de phase global près sous la forme :

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)e^{i\varphi}\,|1\rangle$$

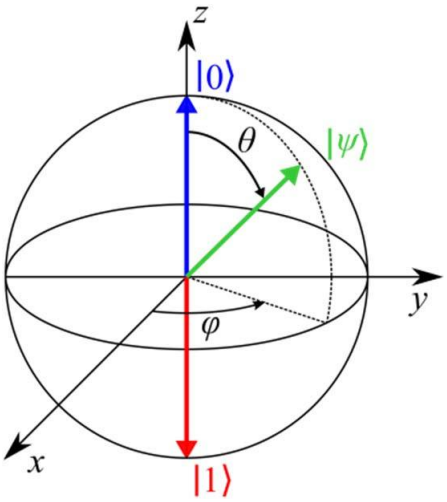


Figure 1 – Sphère de Bloch

avec $\theta \in [0, \pi]$ et $\varphi \in [0, 2\pi]$. On peut alors naturellement associer l'état $|\psi\rangle$ à un point sur une sphère comme l'illustre la figure 1. On appelle cette représentation la **sphère de Bloch**. L'état $|0\rangle$ correspond au pôle nord (de coordonnées cartésiennes $(0, 0, 1)$) de la sphère et l'état $|1\rangle$ au pôle sud (de coordonnées $(0, 0, -1)$).
Définissons enfin deux états particuliers :

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \qquad |-\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

correspondant respectivement aux points de coordonnées $(1, 0, 0)$ et $(-1, 0, 0)$ sur la sphère de Bloch.

Une porte quantique est une transformation linéaire isométrique de l'espace des états dans lui-même. Il s'agit donc d'une matrice U inversible telle que $U^{-1} =^T \overline{U}$. On note communément $U^\dagger =^T \overline{U}$. On détaille plusieurs exemples dans la suite.

3.1.2 Portes de Pauli

On définit ici trois portes quantiques élémentaires :

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Ces trois matrices sont appelées **portes de Pauli**.
La matrice X est appelée **porte NOT**, pour la raison suivante :

$$X|0\rangle = |1\rangle \text{ et } X|1\rangle = |0\rangle$$

De plus, les kets $|+\rangle$ et $|-\rangle$ sont vecteurs propres de X :

$$X|+\rangle = |+\rangle \text{ et } X|-\rangle = -|-\rangle$$

Pour la porte Z, on a :

$$Z\left|0\right\rangle = \left|0\right\rangle \text{ et } Z\left|1\right\rangle = -\left|1\right\rangle$$

Enfin, on connaît aussi les vecteurs propres de la porte Y, ce sont les vecteurs :

$$\left|g\right\rangle = \frac{\left|0\right\rangle + i\left|1\right\rangle}{\sqrt{2}} \qquad \left|d\right\rangle = \frac{\left|0\right\rangle - i\left|1\right\rangle}{\sqrt{2}}$$

On se contente de mentionner que ces états correspondent respectivement aux points de coordonnées $(0,1,0)$ et $(0,-1,0)$ sur la sphère de Bloch. Nous n'en aurons pas besoin dans la suite. Notons enfin que $X^2 = Y^2 = Z^2 = I$.

3.1.3 Portes de rotation

On définit 3 nouvelles portes dites de **rotation** dépendant d'un paramètre angulaire θ :

$$Rx(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad Ry(\theta) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \quad Rz(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$$

Ces portes réalisent géométriquement des rotations sur la sphère de Bloch respectivement autour des axes x, y et z. On a alors naturellement :

$$Rx(\theta)^{-1} = Rx(-\theta) \quad Ry(\theta)^{-1} = Ry(-\theta) \quad Rz(\theta)^{-1} = Rz(-\theta)$$

Remarquons que les portes de Pauli sont des cas particuliers des portes de rotations :

$$Rx(\pi) = -iX \quad Ry(\pi) = -iY \quad Rz(\pi) = -iZ$$

Le facteur $-i$ apparaissant est un facteur de phase global si bien que pour tout $\left|\psi\right\rangle$, le états $X\left|\psi\right\rangle$ et $Rx(\pi)\left|\psi\right\rangle$ seront confondus : ces états sont représentés par le même point sur la sphère de Bloch.
Ces portes de rotations permettent de construire des superpositions quelconques. Par exemple, on a :

$$Ry(\theta)\left|0\right\rangle = \cos(\theta/2)\left|0\right\rangle + \sin(\theta/2)\left|1\right\rangle$$

Ainsi, pour $\lambda \in [0,1]$, on peut contruire l'état $\lambda\left|0\right\rangle + (1-\lambda)\left|1\right\rangle$ (prendre $\theta = 2\arccos\lambda$).
Enfin, les portes de rotations peuvent s'écrire comme des exponentielles de portes de Pauli :

$$Rx(\theta) = \exp\left(-i\frac{\theta}{2}X\right) \quad Ry(\theta) = \exp\left(-i\frac{\theta}{2}Y\right) \quad Rz(\theta) = \exp\left(-i\frac{\theta}{2}Z\right)$$

3.1.4 Porte de Hadamard

La dernière porte agissant sur un qubit seul que l'on a besoin de définir ici est la **porte de Hadamard** :

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Cette porte réalise le changement de base de $(\left|0\right\rangle, \left|1\right\rangle)$ vers $(\left|+\right\rangle, \left|-\right\rangle)$:

$$H\left|0\right\rangle = \left|+\right\rangle \quad H\left|1\right\rangle = \left|-\right\rangle$$

De plus : $H^{-1} = H$. On a donc la relation $X = HZH$.
Cette porte est particulièrement utile pour créer des superpositions équilibrées d'états (typiquement l'état $\left|+\right\rangle$).

3.2 Système de plusieurs qubits

Nous avons décrit ci-dessus l'état d'un qubit seul et les transformations qu'on pouvait lui appliquer. On peut maintenant s'intéresser à un système de N qubits, correspondant à un espace de Hilbert de dimension 2^N d'après les propriétés du produit tensoriel.
Étudions dans un premier temps le cas où on a deux qubits indépendants, le premier dans l'état $|\varphi_0\rangle$ et le second dans l'état $|\varphi_1\rangle$. L'état du système total est alors : $|\varphi\rangle = |\varphi_0\rangle \otimes |\varphi_1\rangle$. Un tel état est dit **factorisé** et semble assez intuitif à appréhender. Au contraire, un état qui ne peut pas se mettre sous cette forme est dit **intriqué**. Ces états sont beaucoup plus contre-intuitifs. Un premier exemple d'état intriqué est l'état dit **de Bell** :

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

On verra dans la suite de cette partie comment construire un tel état.
Si $x_0, .., x_{N-1} \in \{0, 1\}$, et

$$x = \sum_{i=0}^{N-1} x_i 2^i$$

On notera :

$$|x\rangle = |x_0 ... x_{N-1}\rangle = |x_0\rangle \otimes ... \otimes |x_{N-1}\rangle$$

Les $|x\rangle$ pour x entre 0 et $2^N - 1$ forment une base possible d'un système à N qubit : tout état quantique de ce système est une combinaison linéaire normée de ces états de base. C'est-à-dire que pour tout état $|\psi\rangle$ on trouve les λ_x tels que

$$|\psi\rangle = \sum_{x=0}^{2^N-1} \lambda_x |x\rangle \quad \text{avec} \quad \sum_{x=0}^{2^N-1} |\lambda_x|^2 = 1$$

Si on mesure l'état $|\psi\rangle$, on obtient $|x\rangle$ avec probabilité $|\lambda_x|^2$. Immédiatement après la mesure, le système est dans l'état $|x\rangle$ mesuré.

Une porte quantique agissant sur un système de N qubits est une matrice U de taille $2^N \times 2^N$ telle que $U^{-1} = U^\dagger$ où $U^\dagger$ est défini comme précédemment. Si $|\varphi\rangle = |\varphi_0\rangle \otimes |\varphi_1\rangle$ est l'état d'un système de deux qubits et U_0 et U_1 sont deux portes quantiques agissant sur un qubit alors $U_0 \otimes U_1$ définit une porte quantique telle que :

$$(U_0 \otimes U_1) |\varphi\rangle = (U_0 |\varphi_0\rangle) \otimes (U_1 |\varphi_1\rangle)$$

Remarquons que :

$$H^{\otimes N} |0\rangle^{\otimes N} = (H |0\rangle)^{\otimes N} = |+\rangle^{\otimes N} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)^{\otimes N} = \frac{1}{\sqrt{2^N}} \sum_{x=0}^{2^N-1} |x\rangle$$

En effet, en distribuant le produit tensoriel, on fait apparaître une et une seule fois toutes les combinaisons de $|0\rangle$ et $|1\rangle$ de longueur N, ce qui correspond à tous les entiers entre 0 et $2^N - 1$. On note $|s\rangle$ l'état décrit ci-dessus, qui est la superposition uniforme de tous les états de base, dont on se servira dans la suite.

Si on travaille avec N qubits et U est une porte quantique agissant sur un seul qubit, on définit U_k comme la porte U agissant sur le qubit k :

$$U_k = I^{\otimes k} \otimes U \otimes I^{\otimes N-k-1}$$

3.2.1 Portes contrôlées

Donnons nous une porte quantique U agissant sur N qubits. On définit la porte U-contrôlée de la façon suivante :

$$CU = \begin{pmatrix} I_N & 0 \\ 0 & U \end{pmatrix}$$

Cette porte agit sur $N+1$ qubits de la façon suivante : soit $|\psi\rangle$ l'état des N premiers qubits, alors :

$$CU(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |0\rangle \qquad CU(|\psi\rangle \otimes |1\rangle) = (U|\psi\rangle) \otimes |1\rangle$$

Autrement dit, si le dernier qubit, dit **qubit de contrôle**, est dans l'état $|0\rangle$ la porte CU ne fait rien ; si, au contraire, le qubit de contrôle est à $|1\rangle$, on applique la porte U sur les N premiers qubits.

Les portes contrôlées permettent de construire des états intriqués. Par exemple, on peut construire l'état de Bell à l'aide d'une porte CX :

$$\begin{aligned} CX\ H_1 |00\rangle &= CX \left(|0\rangle \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \\ &= \frac{1}{\sqrt{2}} (CX|00\rangle + CX|01\rangle) \\ &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \end{aligned}$$

On peut bien sûr contrôler des portes contrôlées. Définissons $C^0U = U$ et, pour tout $k \geq 0$, $C^{k+1}U = C(C^kU)$. La porte C^kU agit sur $N+k$ qubits ; elle applique la porte U aux N premiers qubits si et seulement si les k qubits de contrôle sont à $|1\rangle$ et ne fait rien dans le cas contraire. Enfin, il est parfois utile de ne vouloir appliquer la porte U que dans le cas où le qubit de contrôle est à $|0\rangle$. Pour cela, on applique simplement une porte X sur le qubit de contrôle avant la porte CU puis une autre porte X après pour annuler l'effet de la première :

$$X_1\ CU\ X_1(|\psi\rangle \otimes |0\rangle) = (U|\psi\rangle) \otimes |0\rangle \qquad X_1\ CU\ X_1(|\psi\rangle \otimes |1\rangle) = |\psi\rangle \otimes |1\rangle$$

Sauf mention explicite du contraire, l'état de contôle sera $|1...1\rangle$. La figure 2 montre les schémas correspondant aux portes contrôlées.

(a) Porte contrôlée par l'état $|1\rangle$

(b) Porte contrôlée par l'état $|0\rangle$

Figure 2 – Représentation d'une porte contôlée

3.2.2 Porte Swap

La dernière porte qui nous sera utile est la porte **swap**. Cette porte agit sur deux qubits et est définie de la façon suivante :

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

On a :

$$S\,|00\rangle = |00\rangle \qquad S\,|01\rangle = |10\rangle \qquad S\,|10\rangle = |01\rangle \qquad S\,|11\rangle = |11\rangle$$

Autrement dit, la porte swap échange les états des deux qubits. Soient $|\psi_0\rangle$ et $|\psi_1\rangle$ les états respectifs de deux qubits. Alors :

$$S(|\psi_0\rangle \otimes |\psi_1\rangle) = |\psi_1\rangle \otimes |\psi_0\rangle$$

La figure 3 montre le schéma d'une telle porte.

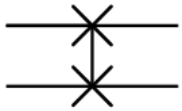


Figure 3 – Représentation d'un porte Swap

3.3 Opérateurs

Un opérateur agissant sur N qubit est simplement une matrice A, généralement symétrique, de taille $2^N \times 2^N$. Pour $|\psi\rangle$ l'état du système, on appelle **valeur moyenne** de A dans l'état $|\psi\rangle$ le nombre $\langle\psi|\,A\,|\psi\rangle$. En particulier, si A est diagonal (ou diagonalisable) :

$$A\,|x\rangle = a_x\,|x\rangle$$

pour les états de base $|x\rangle$ et pour

$$|\psi\rangle = \sum_{x=0}^{2^N-1} \lambda_x\,|x\rangle \qquad \text{avec} \qquad \sum_{x=0}^{2^N-1} |\lambda_x|^2 = 1$$

alors :

$$\langle\psi|\,A\,|\psi\rangle = \sum_{x=0}^{2^N-1} a_x|\lambda_x|^2$$

s'interprète comme une espérance. On dira qu'on **mesure** a_x avec probabilité $|\lambda_x|^2$.

Soit $|\psi\rangle$ l'état quantique d'un système de qubits. L'opérateur $|\psi\rangle\langle\psi|$ est la projection dans la direction $|\psi\rangle$. En effet :

$$(|\psi\rangle\langle\psi|)^2 = (|\psi\rangle\langle\psi|)(|\psi\rangle\langle\psi|) = |\psi\rangle\,(\langle\psi|\psi\rangle)\,\langle\psi| = |\psi\rangle\langle\psi|$$

par associativité du produit matriciel et car $\langle\psi|\psi\rangle = \| |\psi\rangle \|^2 = 1$.

On définit enfin $\hat{x}_k$ agissant sur un état de base $|x\rangle = |x_0...x_{N-1}\rangle$ de la façon suivante :

$$\hat{x}_k |x\rangle = x_k |x\rangle = \begin{cases} 0 & \text{si } x_k = 0 \\ |x\rangle & \text{si } x_k = 1 \end{cases}$$

4 Informatique quantique - illustrations et choix d'un algorithme

4.1 Exemples d'algorithmes quantiques canoniques

Ces éléments étant introduits, nous allons pouvoir présenter succinctement plusieurs algorithmes étudiés dans les premiers temps du projet. Il nous a, en effet, semblé important de développer une connaissance basique des principaux protocoles et algorithmes quantiques avant de nous intéresser à une approche quantique du TSP.
Dans le cadre de cette démarche préliminaire, nous avons ainsi étudié plusieurs algorithmes exploitant notamment le principe de superposition, le principe de réduction du paquet d'onde et le phénomène d'intrication : la transformée de Fourier quantique, l'estimation de phase, et les algorithmes de Shor et de Grover. La suite de ce paragraphe sera consacrée à une description succincte de ces algorithmes.

- La **transformée de Fourier quantique** permet de décomposer une formule en produit de plusieurs matrices unitaires. En utilisant cette méthode de décomposition, la transformée de Fourier discrète peut être utilisée comme un circuit quantique, qui comprend plusieurs portes Hadamard et des portes de rotation.
- L'**algorithme d'estimation de phase** quantique permet d'estimer la valeur propre d'un opérateur unitaire associée à un vecteur propre donné. Soit $|\phi\rangle$ un vecteur propre de $\mathbf{U}$, on a donc $\mathbf{U}|\phi\rangle = e^{2\pi i\theta}|\phi\rangle$, car les valeurs propres d'un opérateur unitaire sont de module 1. L'algorithme est basé sur cette égalité et permet alors de trouver la valeur de la phase θ. Plus précisément, on dispose t qubits de registres de contrôle initialisés à zéro $|0...0\rangle$ et un qubit dans l'état $|\phi\rangle$. Puis on applique t-bit portes de Hadamard aux qubits de contrôle pour les superposer, la superposition des états donnant : $\frac{1}{2^{\frac{t}{2}}}(|0\rangle + |1\rangle)^{\otimes n}$. On applique ensuite l'opérateur unitaire sur le registre cible uniquement si son qubit de contrôle correspondant est $|1\rangle$. L'état $|\phi\rangle$ doit rester inchangé après cette opération. Après une transformation de Fourier inverse, en mesurant les registres de contrôle, on obtient une chaîne de nombres binaires. Ce dernier divisé par 2^t sera la phase de l'état $|\phi\rangle$ que l'on cherche à estimer [Com].
- L'**algorithme de Shor** permet quant à lui de factoriser en nombres premiers un entier naturel N en temps $\mathcal{O}((\log N)^3)$, et pourrait bouleverser le monde de la cryptographie en rendant le chiffrement RSA inefficace lorsque des ordinateurs quantiques assez puissants pour l'implémenter apparaîtront.

— L'**algorithme de Grover** sert à chercher des éléments qui répondent à un critère donné parmi N éléments non classés. Ce problème nécessite $\mathcal{O}(N)$ tests dans les calculs classiques car, dans le pire des cas, il faut tester tous les éléments. L'algorithme quantique a, quant à lui, une complexité de $\mathcal{O}(\sqrt{N})$.

Les algorithmes ci-dessus ont de nombreuses applications et occupent une position centrale dans la construction d'autres algorithmes quantiques plus complexes.

4.2 Sélection d'un algorithme quantique pour la résolution du TSP

Après les phases de découverte plus approfondies du TSP et de l'algorithmique quantique, nous nous sommes plongés dans la littérature existante sur les alliances entre ces deux derniers points. Par tâtonnements, nous avons regardé les pistes déjà explorées, et eu un aperçu du large panel des méthodes existantes. Face aux nombreux défis conceptuels que ces dernières nous posaient, nous devions faire un choix sur une petite quantité de méthodes sur lesquelles se concentrer pour pouvoir avancer de manière efficace.

Ce choix s'est imposé par des coïncidences fructueuses. À la suite d'une PC de physique quantique avancée et d'une discussion avec Mme Senellart, professeure chargée de la PC, nous avons pu entrer en contact avec un chercheur du plateau de Saclay, M. Shane Mansfield, travaillant sur une technique qui nous semblait intéressante et abordable avec notre niveau de connaissances : le QAOA [FGG14], pour **Quantum Approximate Optimization Algorithm**.

Nous l'étudierons en détail dans la partie suivante et en présenterons différentes implémentations dans les parties 6 et 7.

5 Le QAOA

Le QAOA est un type d'algorithme de minimisation sous contraintes. L'idée de ce type d'algorithme est de partir de l'état fondamental d'un certain opérateur puis de transformer cet opérateur en celui modélisant notre problème, de sorte qu'à la fin de la transformation le système soit dans l'état fondamental de cet opérateur "cible". Pour expliquer le fonctionnement du QAOA, nous allons d'abord expliquer celui d'un algorithme voisin : le QAA, pour **Quantum Adiabatic Algorithm**.

5.1 QAA

Le QAA est un algorithme qui, comme le QAOA, a pour but de réaliser une minimisation sous contraintes. L'idée est également de transformer un certain état de départ, pris comme l'état fondamental d'un opérateur bien choisi, en un état solution du problème considéré - c'est à dire minimisant l'énergie de l'opérateur associé à ce problème. Pour ce faire, l'opérateur initial est modifié suffisamment lentement vers l'opérateur modélisant le problème. Ce faisant, le système reste dans son état d'énergie minimal à la condition qu'à chaque instant il y ait un écart entre la valeur propre minimale et les autres valeurs propres du système. Ce résultat qui semble intuitif se base sur le théorème adiabatique que nous allons présenter plus précisément.

5.1.1 Théorème adiabatique

Considérons un hamiltonien dépendant du temps $H(t)$, que nous faisons évoluer durant un temps T. Si notre système est dans l'état $|\psi\rangle$ à l'origine, alors son évolution est déterminée par l'équation de Schrödinger :

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle$$

Notons $|\psi^H(t)\rangle$ un état propre instantané de $H(t)$ avec $E^H(t)$ sa valeur propre associée. Enfin, notons $|\psi(t)\rangle$ l'état du système à l'instant t. Nous choisissons au temps $t = 0$ notre système $|\psi(0)\rangle$ comme un état propre de l'hamiltonien $H(0)$: $|\psi(0)\rangle = |\psi^H(0)\rangle$ est l'état initial.

Le théorème adiabatique établit que, s'il existe un saut spectral $\lambda > 0$ autour de $E^H(t)$ pour tout $t \in [0, T]$, alors pour T assez grand (typiquement $T = O(\lambda^{-2})$ [Zho+20]) l'état final $|\psi(T)\rangle$ peut être choisi arbitrairement proche de l'état propre $|\psi^H(T)\rangle$ de $H(T)$. On pourra consulter [AR06] pour une démonstration détaillée.

Ce mécanisme permet d'envisager un algorithme qui déterminerait l'état fondamental d'un hamiltonien final H_f correspondant à un problème d'optimisation à partir de celui d'un hamiltonien initial H_i plus facile à construire. Le QAA [Far+00] considère ainsi l'hamiltonien :

$$H(t) = \frac{T - t}{T} H_i + \frac{t}{T} H_f$$

On a alors $H(0) = H_i$ et $H(T) = H_f$. En supposant que pour un tel hamiltonien le saut spectral λ correspondant à l'état de plus basse énergie soit non nul, et que l'évolution adiabatique soit

respectée (i.e. le temps d'évolution soit suffisamment long), alors l'état obtenu en sortie sera bien solution du problème d'optimisation considéré d'après le théorème adiabatique.

Une telle évolution continue de l'hamiltonien $H(t)$ n'étant pas directement implémentable sur un support digital (fonctionnant avec des portes), il faut l'approcher via une discrétisation temporelle pour pouvoir l'appliquer. En effet, on peut simuler l'action d'un hamiltonien constant H pendant un temps Δt en implémentant un circuit approximant l'opérateur unitaire $\exp(-i\Delta t H)$. C'est ce que nous allons à présent détailler.

5.1.2 Trotterisation

Nous allons ici justifier que l'évolution sous l'hamiltonien

$$H(t) = \frac{T-t}{T} H_i + \frac{t}{T} H_f$$

peut être réalisée par N applications successives des opérateurs $(T-t)/T\ H_i$ et $t/T\ H_f$ pendant des temps $\Delta t = T/N$.
Pour formuler correctement ce résultat, nous allons le traduire en terme d'opérateurs d'évolutions $U(t', t)$. Cet opérateur traduit l'évolution du système d'un temps t à un temps t' : $|\psi(t')\rangle = U(t', t)\,|\psi(t)\rangle$. D'après l'équation de Schrödinger, on a

$$i\hbar \frac{d}{dt'} U(t', t) = H(t') U(t', t)$$

Dans la suite, nous ferons abstraction du terme $\hbar$ ce qui peut être fait en le rajoutant dans le terme H. Avec ces notations introduites, nous cherchons à montrer que :

$$U(T, 0) \approx \prod_{k=0}^{N-1} \exp\left(-i\Delta t \left(\frac{T - k\Delta t}{T} H_i\right)\right) \exp\left(-i\Delta t \left(\frac{k\Delta t}{T} H_f\right)\right)$$

Pour expliquer ce résultat, nous allons suivre l'exemple de [Hab]. Tout d'abord, on applique la formule de Taylor en $\Delta t'$ à $U(t' + \Delta t', t)$, on obtient ainsi :

$$U(t' + \Delta t', t) = U(t', t) + \frac{d}{dt'} U(t', t)\Delta t' + O(\Delta t'^2) = U(t', t) - iH(t')U(t', t)\Delta t' + O(\Delta t'^2)$$

D'où, pour $t = t'$:

$$U(t + \Delta t, t) = I - iH(t)\Delta t + O(\Delta t^2) = \exp(-i\Delta t H(t)) + O(\Delta t^2)$$

Par suite :

$$\begin{aligned} U(T, 0) &= \prod_{k=0}^{N-1} U((k+1)\Delta t, k\Delta t) \\ &= \prod_{k=0}^{N-1} \left(\exp(-i\Delta t\, H(k\Delta t)) + O(\Delta t^2)\right) \\ &= \prod_{k=0}^{N-1} \exp(-i\Delta t\, H(k\Delta t)) + O(\Delta t^2) \end{aligned}$$

D'où en passant à la limite :

$$U(T,0) = \lim_{N\to+\infty} \prod_{k=0}^{N-1} \exp(-i\Delta t\, H(k\Delta t))$$

Les opérateurs $H(t)$ et $H(t')$ ne commutent *a priori* pas. Dans le cas où ils commuteraient pour tout $t, t' \in [0,T]$, le résultat se déduirait immédiatement. En effet, on aurait alors :

$$U(T,0) = \lim_{N\to+\infty} \exp\left(-i\frac{T}{N}\sum_{k=0}^{N-1} H(k\Delta t)\right) = \exp\left(-i\int_0^T H(t)dt\right)$$

Ceci signifie bien que notre approximation est correcte, on peut simuler l'évolution par N applications successives pendant Δt de l'hamiltonien pris aux temps $k\Delta t$.
Pour continuer la réécriture de $U(T,0)$ dans le cas non-commutatif, nous allons nous appuyer sur [Hab] en repartant de l'équation différentielle :

$$U(t',0) = I - i\int_0^{t'} H(t)U(t,0)dt$$

On peut alors écrire $U(t',0)$ sous la forme :

$$U(t',0) = I + \sum_{n=1}^{+\infty} (-i)^n \int_0^{t'} dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n H(t_1)H(t_2)...H(t_n)$$

Pour pouvoir simplifier cette expression, nous introduisons l'opérateur d'ordonnancement temporel défini, pour $t_{k_1} \geq t_{k_2} \geq ... \geq t_{k_n}$, par :

$$\mathcal{T}[H(t_1)H(t_2)...H(t_n)] = H(t_{k_1})H(t_{k_2})...H(t_{k_n})$$

Grâce à cet opérateur, une intégrale de la forme

$$J_n = \int_0^{t'} dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n H(t_1)H(t_2)...H(t_n)$$

se réécrit

$$J_n = \frac{1}{n!}\int_0^{t'} dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n \mathcal{T}[H(t_1)H(t_2)...H(t_n)]$$

Ainsi,

$$\begin{aligned} U(t',0) &= I + \sum_{n=1}^{+\infty} (-i)^n \int_0^{t'} dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n H(t_1)H(t_2)...H(t_n) \\ &= I + \sum_{n=1}^{+\infty} \frac{(-i)^n}{n!} \int_0^{t'} dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n \mathcal{T}[H(t_1)H(t_2)...H(t_n)] \\ &:= \mathcal{T}\exp\left(-i\int_0^{t'} H(t)dt\right) \end{aligned}$$

Cette exponentielle temporellement ordonnée $\mathcal{T}\exp$ peut être approchée en utilisant la formule de Trotter-Suzuki. Dans sa plus simple expression, cette dernière établit que pour A et B des opérateurs quelconques et x un paramètre :

$$\exp(x(A+B)) = \exp(xA)\exp(xB) + O(x^2)$$

On peut ainsi approcher l'opérateur d'évolution, toujours avec $\Delta t = T/N$ [Sun+18] :

$$U(T,0) = \mathcal{T}\exp(-i\int_0^T H(t)dt) \approx \prod_{k=0}^{N-1} \exp(-iH(k\Delta t)\Delta t)$$

En injectant la formule de l'hamiltonien du QAA, on obtient ainsi :

$$U(T,0) \approx \prod_{k=0}^{N-1} \exp\left(-i[\frac{k\Delta t}{T}H_i + \frac{T-k\Delta t}{T}H_f]\Delta t\right)$$

Enfin, en utilisant à nouveau la formule de Trotter-Suzuki :

$$U(T,0) \approx \prod_{k=0}^{N-1} \exp\left(-i\Delta t\left(\frac{T-k\Delta t}{T}H_i\right)\right)\exp\left(-i\Delta t\left(\frac{k\Delta t}{T}H_f\right)\right)$$

Cette approximation est exacte en passant à la limite pour $N \longrightarrow +\infty$, et permet donc la simulation souhaitée.
En conclusion, nous avons justifié la correction de la simulation par décomposition de Trotter-Suzuki du QAA, grâce à laquelle nous démontrerons celle du QAOA une fois ce dernier introduit.

5.2 Application au QAOA

Le principe de fonctionnement du QAOA est analogue à celui du QAA. Pour rappel, il consiste à partir de l'état d'énergie minimale d'un hamiltonien et à le transformer afin d'obtenir un état d'énergie minimale pour l'hamiltonien final. La différence entre ces deux algorithmes réside dans la sélection des pas, qui sont tous identiques et choisis suffisamment petits dans le cas du QAA, alors qu'ils sont déterminés par un processus d'optimisation dans le cas du QAOA. La figure 4 issue de [VBB19] permet d'illustrer visuellement ces différences d'évolution entre QAA et QAOA.

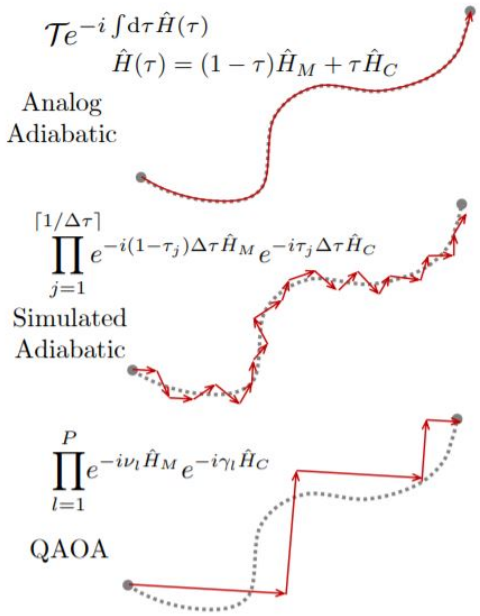


Figure 4 – Différence entre le QAA et le QAOA

Le premier cas correspond à l'évolution adiabatique continue de l'hamiltonien décrite dans la partie précédente par l'exponentielle temporellement ordonnée, et qui n'est pas réalisable sur un support non analogique, i.e un support de type numérique fonctionnant avec des portes quantiques. Le second présente la simulation de cette évolution via la décomposition de Trotter-Suzuki, et qui correspond à la mise en oeuvre pratique du QAA. Le dernier illustre le fonctionnement du QAOA, qui autorise des évolutions rapides quand elles permettent de meilleurs résultats.

5.2.1 Fonctionnement du QAOA

Avant de traiter le choix par optimisation des pas d'évolution de l'algorithme, nous allons commencer par définir plus précisément les hamiltoniens initiaux et finaux, qui seront désormais qualifiés respectivement d'hamiltoniens de conduite et de coût. Pour cela, nous formalisons le

problème d'optimisation sous contrainte que nous souhaitons résoudre de la façon suivante :

Soient $m \in \mathbb{N}^*$ clauses α, auxquelles on associe les fonctions C_α définies par $C_\alpha(x) = 0$ si $x \in \{0,1\}^n$ satisfait la clause α et 1 (ou tout autre nombre strictement positif) sinon.
On cherche à minimiser la fonction :

$$C(x) = \sum_{\alpha=1}^{m} C_\alpha(x)$$

Construction de l'hamiltonien de coût

On peut associer à chaque C_α un opérateur $\hat{C}_\alpha$:

$$\hat{C}_\alpha = \sum_x C_\alpha(x) \left|x\right\rangle \left\langle x\right|$$

On a donc pour tout x :

$$\hat{C}_\alpha \left|x\right\rangle = C_\alpha(x) \left|x\right\rangle$$

On construit ainsi l'hamiltonien de coût dont on cherchera à obtenir un état propre d'énergie minimale :

$$\hat{C} = \sum_{\alpha=1}^{m} \hat{C}_\alpha$$

On fera par la suite correspondre la fonction C avec l'opérateur $\hat{C}$.

Choix de l'hamiltonien de conduite

Comme présenté dans l'exposition de la démarche générale du QAOA, il y a une grande liberté dans la sélection de l'hamiltonien du moment qu'un saut spectral sépare la première et la deuxième valeur propre de l'hamiltonien instantané tout au long de l'évolution.
On choisit donc un opérateur ayant l'avantage d'avoir un état fondamental simple à construire avec un ordinateur quantique, et qui ait une composante non nulle sur chacun des états possibles de notre problème. L'hamiltonien de conduite classiquement adopté est ainsi le suivant :

$$B = -\sum_{i=0}^{N-1} X_i$$

Son état fondamental est la superposition uniforme $\left|s\right\rangle$ définie précédemment. On a :

$$B \left|s\right\rangle = \left(-\sum_{k=0}^{N-1} X_k\right) \left|+\right\rangle^{\otimes N} = -\sum_{k=0}^{N-1} \left(\left|+\right\rangle^{\otimes N}\right) = -N \left|s\right\rangle$$

Nous proposerons d'autres choix plus loin.
Comme pour le QAA, l'objectif est maintenant de faire évoluer l'hamiltonien du système de l'opérateur B à l'opérateur C. Pour rappel, l'évolution choisie dans le QAA est :

$$H(t) = \left(1 - \frac{t}{T}\right) B + \frac{t}{T} C$$

avec T le temps total d'évolution du système.
Pour réaliser l'implémentation sur machine, le temps T doit être discrétisé en N pas de temps Δt, de sorte que les opérateurs d'évolution temporelle sont appliqués successivement avec des pas de temps Δt.
On pose :

$$U(C,\gamma) = e^{-i\gamma C} \qquad U(B,\beta) = e^{-i\beta B}$$

L'état final est alors donné par :

$$U(B,\Delta t)U(C,\Delta t)...U(B,\Delta t)U(C,\Delta t)\,|s\rangle$$

L'idée du QAOA est d'optimiser cette discrétisation du temps en autorisant des Δt différents les uns des autres et suffisamment grands pour permettre des évolutions diabatiques si elles améliorent la qualité d'approximation de l'algorithme. Pour ce faire, nous introduisons l'hyperparamètre p représentant le nombre de cycles d'application de B et C. On notera $\boldsymbol{\gamma} = \gamma_1...\gamma_p$ et $\boldsymbol{\beta} = \beta_1...\beta_p$ les temps associés à chacun des opérateurs. Au premier cycle, on applique donc l'opérateur C pendant un temps γ_1, puis l'opérateur B pendant un temps β_1, avant de réitérer l'opération pour la faire p fois au total. Enfin, en notant $|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$ l'état final du système après évolution de $|s\rangle$, on obtient :

$$|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle = U(B,\beta_p)U(C,\gamma_p)...U(B,\beta_1)U(C,\gamma_1)\,|s\rangle$$

La figure 5 résume schématiquement le fonctionnement du QAOA.

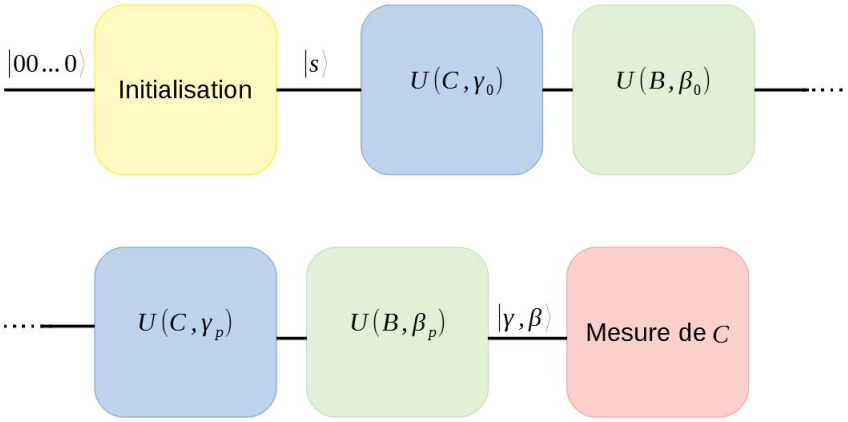


Figure 5 – Circuit utilisé pour le QAOA

Pour modéliser efficacement l'évolution associée à H, l'objectif est maintenant de choisir judicieusement les paramètres $\boldsymbol{\beta}$ et $\boldsymbol{\gamma}$, c'est-à-dire de telle sorte qu'ils minimisent la valeur moyenne de l'opérateur de coût $\langle\boldsymbol{\gamma},\boldsymbol{\beta}|\,C\,|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$. On cherche ainsi à minimiser la fonction :

$$F_p(\boldsymbol{\gamma},\boldsymbol{\beta}) = \langle\boldsymbol{\gamma},\boldsymbol{\beta}|\,C\,|\boldsymbol{\gamma},\boldsymbol{\beta}\rangle$$

Pour ce faire, on réalise une optimisation avec un algorithme classique. En posant

$$M_p = \min_{\gamma,\beta} F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$$

la correction du QAOA est équivalente à la convergence suivante :

$$\lim_{p\to\infty} M_p = \min_x C(x)$$

Sa démonstration fera l'objet de la partie suivante.

5.2.2 Justification du QAOA

On note que par définition :

$$\forall p \geq 1, M_p \geq \min_x C(x)$$

Par ailleurs :

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U(B, \beta_p)U(C, \gamma_p)...U(B, \beta_1)U(C, \gamma_1)\,|s\rangle = \prod_{k=1}^{p} \exp(-i\beta_k B)\exp(-i\gamma_k C)\,|s\rangle$$

Autrement dit, avec $T^{\gamma,\beta} = \sum_{i=1}^{p}(|\gamma_i| + |\beta_i|)$:

$$U_{QAOA}^{\gamma,\beta}(T^{\gamma,\beta}, 0) = \prod_{k=0}^{p} \exp(-i\beta_k B)\exp(-i\gamma_k C)$$

On fixe maintenant $T \geq 0$ vérifiant les conditions du théorème adiabatique. En utilisant l'approximation de $U(T, 0)$ trouvée dans la partie 5.1.2 on peut donc choisir, pour tout p suffisamment grand, $(\gamma_{QAA,p}, \beta_{QAA,p}) \in \mathbb{R}^p \times \mathbb{R}^p$ tels que $U_{QAOA}^{\gamma_{QAA,p},\beta_{QAA,p}}(T, 0)$ soit arbitrairement proche de $U(T, 0)$. Par suite :

$$U_{QAOA}^{\gamma_{QAA,p},\beta_{QAA,p}}(T, 0) \underset{p\to+\infty}{\longrightarrow} U(T, 0)$$

Ainsi :

$$F_p(\gamma_{QAA}, \beta_{QAA}) \underset{p\to+\infty}{\longrightarrow} \langle s|\, U(T, 0)^\dagger C U(T, 0)\,|s\rangle$$

Donc, d'après le théorème adiabatique :

$$F_p(\gamma_{QAA,p}, \beta_{QAA,p}) \underset{p\to+\infty}{\longrightarrow} \min_x C(x)$$

Or, à p fixé :

$$M_p \leq F_p(\gamma_{QAA,p}, \beta_{QAA,p})$$

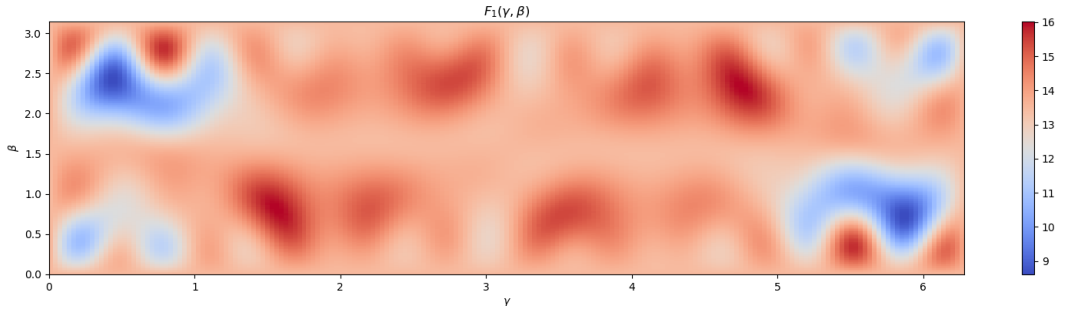
D'où le résultat.

5.3 Symétries de la fonction F_p

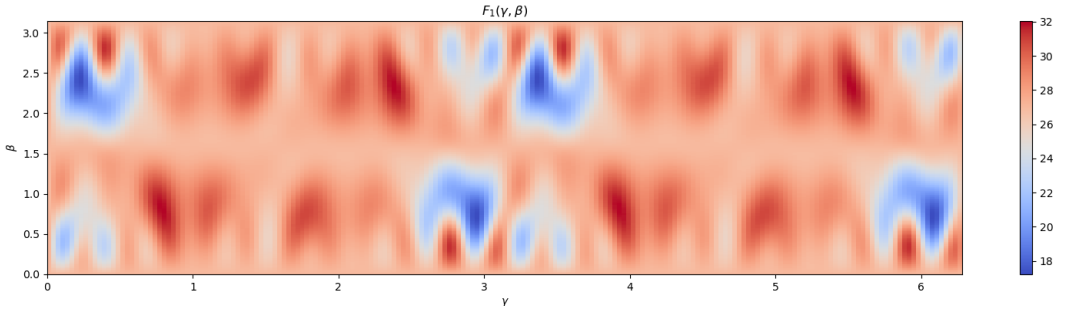
Les paramètres $\boldsymbol{\gamma} = (\gamma_k)_{1\leq k\leq p}$ et $\boldsymbol{\beta} = (\beta_k)_{1\leq k\leq p}$ sont *a priori* à chercher dans l'espace tout entier $\mathbb{R}^p \times \mathbb{R}^p$. Cependant, en étudiant les symétries de la fonction F_p, on peut se restreindre à des domaines plus petits.
D'abord, remarquons que comme C et B sont à coefficients réels : $U(B, -\beta) = \overline{U(B, \beta)}$ et $U(C, -\gamma) = \overline{U(C, \gamma)}$ donc si l'état initial est à coefficients dans $\mathbb{R}$:

$$|-\boldsymbol{\gamma}, -\boldsymbol{\beta}\rangle = \overline{|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle}$$

d'où $F_p(-\boldsymbol{\gamma}, -\boldsymbol{\beta}) = \overline{F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})} = F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$. La fonction F_p est donc paire et on peut se passer de la moitié de l'espace. Autrement dit, on peut se restreindre à chercher les γ_k dans $\mathbb{R}$ et les β_k dans $\mathbb{R}_+$.
De plus, si les valeurs propres de C et B sont entières, on a en : $U(C, \gamma + 2\pi) = U(C, \gamma)$ et $U(B, \beta + 2\pi) = U(B, \beta)$. Ainsi, la fonction F_p est 2π-périodique dans toutes les directions. On peut donc se restreindre à chercher $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ dans $[0, 2\pi]^p \times [0, 2\pi]^p$ et donc dans $[0, 2\pi]^p \times [0, \pi]^p$ avec la remarque précédente. En pratique, C ne prendra que des valeurs entières car on considère uniquement des instances du TSP avec des poids entiers. Pour ce qui est de B, l'opérateur proposé plus haut vérifie bien cette propriété mais on verra dans la suite un autre choix pour B qui n'est pas à valeurs propres entières.
Enfin, si les valeurs propres de C ont encore plus de régularité, par exemple si leur *pgcd* est strictement plus grand que 1, la périodicité dans les directions $\boldsymbol{\gamma}$ est encore plus grande. En effet, si tous les coûts sont multiples d'un entier a alors les γ_k seront à chercher dans $[0, 2\pi/a]$ comme l'illustre la figure 6.



(a) Fonction F_1 pour des coûts premiers entre eux



(b) Fonction F_1 pour des coûts doublés

Figure 6 – Illustration de la périodicité de F_1

5.4 Similitudes et différences entre le QAA et le QAOA

La partie précédente pourrait conduire à comprendre le QAOA comme une simple optimisation des coefficients du QAA, ce qui n'est pas le cas. En effet, dans le QAOA, l'évolution n'est plus forcément adiabatique, l'état sur lequel nous travaillons pouvant changer de niveau d'énergie, ce qui n'était pas autorisé dans le QAA.
De plus, en reprenant les définitions de $M_p = \min_{\gamma,\beta} F_p(\boldsymbol{\gamma}, \boldsymbol{\beta})$ et $F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}| \, C \, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$, on observe que :

$$\forall p \geq 1, M_p \geq M_{p+1}$$

Ainsi, la qualité d'approximation du QAOA est croissante en p, ce qui n'est pas le cas du QAA. Plus précisément, la référence [Cro+14] donne un exemple où la probabilité de succès du QAA n'est absolument par croissante en le temps d'évolution, comme en témoigne la figure 7 extraite de cet article :

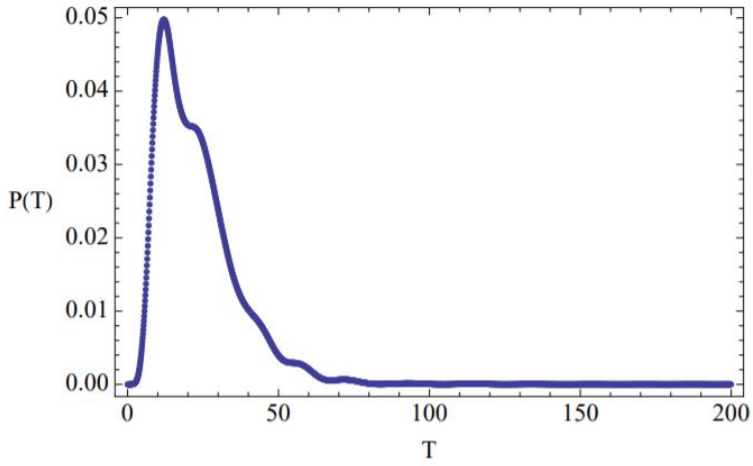


Figure 7 – Un exemple de QAA

La différence entre QAOA et QAA est en fait d'autant plus manifeste que le saut spectral autour de l'état fondamental est réduit, l'avantage principal du QAOA étant alors de pouvoir profiter de transitions diabatiques. Pour s'en convaincre, on peut s'intéresser à la comparaison entre ces deux algorithmes effectuée dans [Zho+20] et que nous allons reprendre en partie. Pour le comparer au QAA, on commence par lisser l'évolution du QAOA en construisant le chemin suivant avec $T = \sum_{i=1}^{p}(|\gamma_i| + |\beta_i|)$:

$$H_{QAOA}(t) = -[f(t)C + (1 - f(t))B]$$

$$f\left(t_k = \sum_{j=1}^{k}(|\gamma_j^*| + |\beta_j^*|) - \frac{1}{2}(|\gamma_k^*| + |\beta_k^*|)\right) = \frac{\gamma_k^*}{|\gamma_k^*| + |\beta_k^*|}$$

On s'intéresse à une situation analogue à celle définie par la figure 8, extraite de [Far+00]. Cette figure représente l'évolution des deux premières valeurs propres au cours de l'évolution de l'hamiltonien.

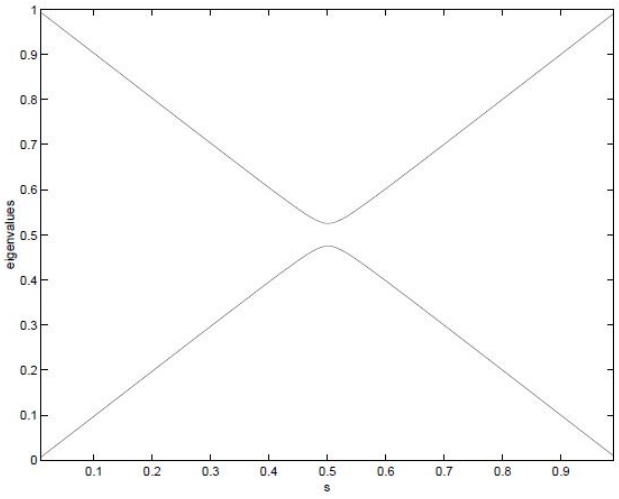
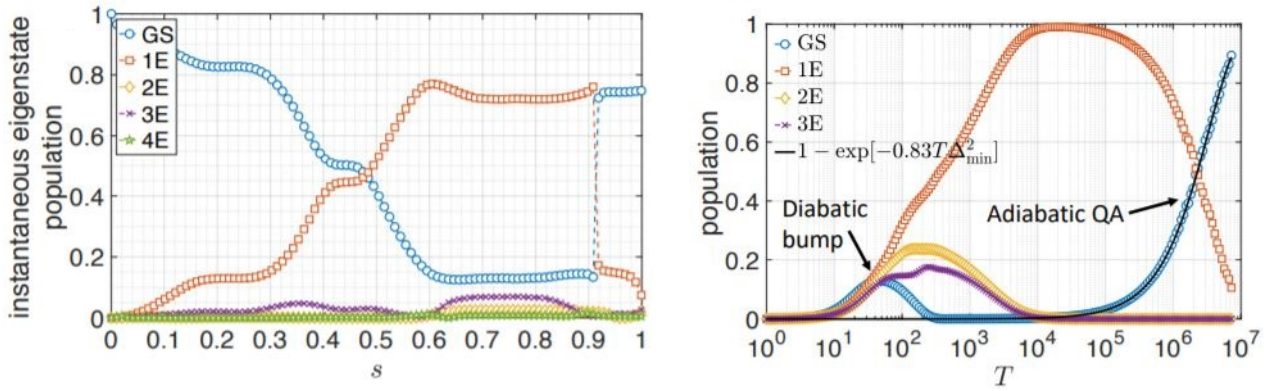


Figure 8 – Un exemple où le saut entre les deux première valeurs propres est petit

On peut alors observer une transition diabatique, en $s \approx 0.92$, correspondant au point de rapprochement maximum de l'état fondamental (GS) et du premier état excité (1E) [fig. 9 (a)], à comparer avec l'évolution adiabatique du QAA sur la même instance [fig. 9 (b)] :



(a) (b)

Figure 9 – Evolutions diabatiques et adiabatiques, extrait de [Zho+20]

Pour une évolution adiabatique, la formule de Landau-Zener donne une estimation théorique de la population dans l'état fondamental : $P_{fond} = 1 - \exp(-cT\lambda^2)$, où c est un paramètre d'ajustement ici égal à 0.83.
Cette prédiction correspond bien aux simulations, même si elle ne rend pas compte des phénomènes diabatiques ("**diabatic bump**") apparaissant pour de faibles valeurs de T et rappelant ceux illustrés par la figure [Cro+14].

L'application directe du QAOA pour différentes valeurs de p permet également de retrouver cette inversion de population, comme le montre la figure 10 :

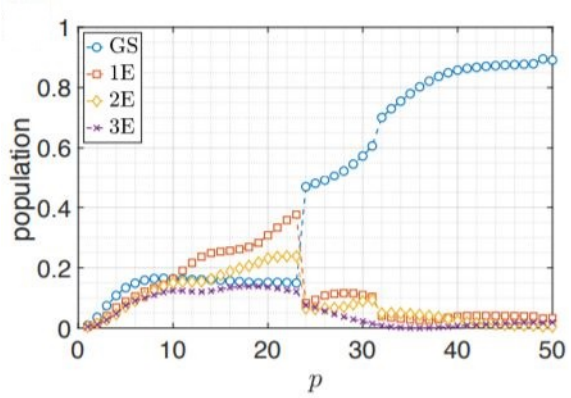


Figure 10 – Evolution diabatique du QAOA, extrait de [Zho+20]

Ces spécificités nous confortent ainsi dans le choix du QAOA comme algorithme pour réaliser nos implémentations.

6 Historique des implémentations

6.1 Approche "naïve"

6.1.1 Cadre théorique

Dans un premier temps, l'étude de [RMS19] et [HD19] nous a fourni les bases d'une application du QAOA au problème du voyageur de commerce. Les deux points principaux de cette implémentation sont la manière dont les chemins sont représentés et la construction des hamiltoniens de coût et de conduite évoqués précédemment.

Une représentation naturelle d'un chemin reliant n sommets nécessite $N = n^2$ qubits : dans la chaîne

$$x = x_{0,0}x_{0,1}...x_{0,n-1}.....x_{n-1,0}...x_{n-1,n-1}$$

le $nt + k$-ème qubit $x_{t,k}$ vaut 1 si et seulement si le chemin passe par le sommet k à l'instant t. On notera $\hat{x}_{t,k} = \hat{x}_{nt+k}$

Bien sûr, toutes les chaînes binaires x ne correspondent pas à des chemins hamiltoniens sur notre graphe. Il faut donc s'assurer que ces chemins non corrects soient de grande énergie, afin de les pénaliser fortement. il faut également pénaliser (moins fortement) les chemins en fonction de leur longueur, afin de favoriser les chemins les plus courts.

1. A chaque instant t, le chemin passe par un et un seul sommet. Cela peut se traduire par l'hamiltonien
$$\mathcal{H}_1 = \sum_{t=0}^{n-1} \left(1 - \sum_{k=0}^{n-1} \hat{x}_{t,k}\right)^2$$
On vérifie en effet que $\langle x| \mathcal{H}_1 |x\rangle = 0$ si $|x\rangle$ vérifie la contrainte et $\langle x| \mathcal{H}_1 |x\rangle > 0$ sinon, la valeur exacte dépendant du nombre de fois que la contrainte est violée. En effet, le terme entre parenthèses correspond à l'écart entre le nombre de sommets visités à l'instant t et 1, qui est la valeur souhaitée.
2. Tous les sommets doivent être visités une et une seule fois. Cela se traduit par :
$$\mathcal{H}_2 = \sum_{k=0}^{n-1} \left(1 - \sum_{t=0}^{n-1} \hat{x}_{t,k}\right)^2$$
Comme précédemment, le terme entre parenthèses correspond à l'écart du nombre de fois que le sommet k est visité à 1, qui est la valeur souhaitée.
3. Les chemins de longueur importante doivent être pénalisés, ce qui fait intervenir les termes de la matrice de coût W du graphe :
$$\mathcal{H}_3 = \sum_{t=0}^{n-1} \left(\sum_{k=0}^{n-1}\sum_{k'=0}^{n-1} W_{k,k'}\hat{x}_{t,k}\hat{x}_{t+1,k'}\right)$$

Il faut alors ajuster la matrice de coût de sorte que les états ne correspondant pas à des chemins soient pénalisés plus fortement que les autres chemins, même les plus longs.

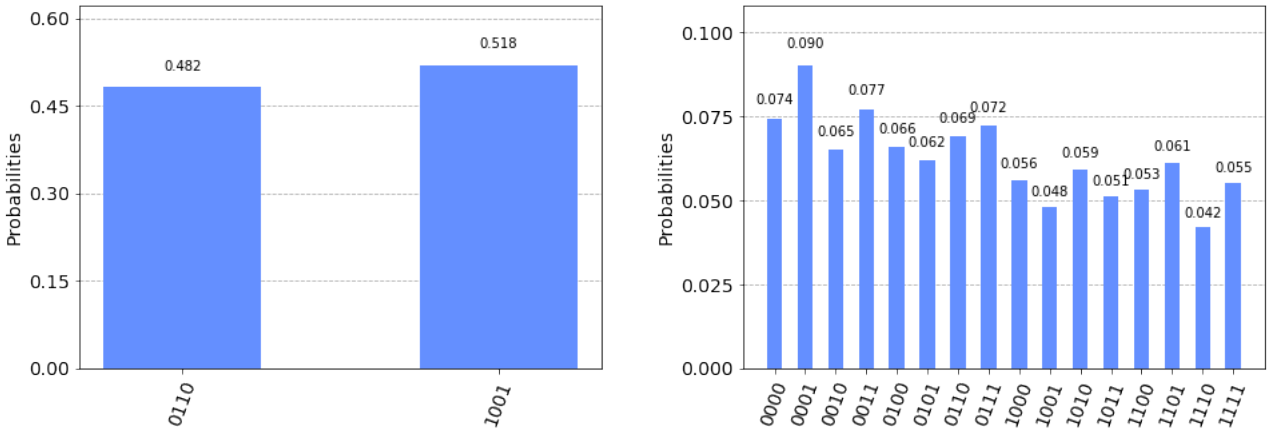
En rassemblant ces contraintes, on obtient notre hamiltonien de coût

$$C = a\mathcal{H}_1 + b\mathcal{H}_2 + c\mathcal{H}_3$$

où a, b et c sont des coefficients représentant l'importance qu'on accorde aux différents hamiltoniens. On a *a priori* $a \sim b > nc|W|$ où on note $|W|$ la valeur maximale de W : on pénalise fortement les chaînes invalides.

6.1.2 Exécution et résultats obtenus

Nous avons donc implémenté sur Qiskit le QAOA tel que décliné ci-dessus. On génère un petit graphe aléatoire ($n = 3$ à cause du nombre de qubits très restreint que proposent les simulateurs quantiques), pour lequel on obtient la solution exacte du TSP par force brute (en essayant toutes les possibilités). On construit ensuite les hamiltoniens de contrainte et de conduite, puis on minimise l'espérance du ket $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$. On construit alors le circuit en trois blocs : préparation de l'état initial, transformation en l'état $|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle$ et mesure des qubits.



(a) Exécution sur un simulateur quantique

(b) Exécution sur une machine réelle

Figure 11 – Résultats obtenus pour n=2

Il apparaît que l'algorithme développé jusqu'ici peut discriminer les états ne représentant pas des chemins valides quand on l'exécute sur un simulateur quantique, mais ne fonctionne pas de manière satisfaisante (même pour $n = 2$) sur un véritable ordinateur quantique à cause du bruit et du temps de cohérence trop faible comme le montre la figure 11.

6.2 Algorithme optimisé - représentation par arêtes

6.2.1 Cadre théorique

Malgré la simplicité de la représentation à n^2 qubits exposée ci-dessus, on se rend vite compte qu'elle est loin d'être optimale. Considérons par exemple le cas $n = 3$: le chemin $0 \to 1 \to 2 \to 0$ est représenté par plusieurs chaînes : 100010001 mais aussi 010001100 et 001100010, représentant respectivement les chemins $1 \to 2 \to 0 \to 1$ et $2 \to 0 \to 1 \to 2$. De plus comme le problème considéré est symétrique, les 3 chaînes correspondant à $2 \to 1 \to 0 \to 2$ représentent aussi ce même chemin. On a donc au total 6 états différents représentant le même chemin. En général, pour n quelconque, chaque chemin admet $2n$ représentations.
Il apparaît donc raisonnable de penser qu'on peut trouver une meilleure représentation des chemins qui permettrait d'une part de réduire le nombre de qubits utilisés mais aussi de concentrer la probabilité éparpillée entre toutes les représentations d'un chemin en un seul état.

On utilise donc la représentation suivante proposée dans [Rua+20] : un chemin sera maintenant donné par l'ensemble des arêtes qu'il emprunte. On considère les chaînes de la forme

$$x = (x_e)_{e\in E} = (x_{j,j'})_{0\leq j<j'<n} = (x_k)_{0\leq k<N}$$

où E est l'ensemble des arêtes du graphe dont on rappelle qu'il est complet.

Le choix de numérotation étant arbitraire, on décide de faire correspondre l'arête (j, j') à $k = \frac{j(j-1)}{2} + j'$, ce qui se révélera utile pour l'implémentation finale qui gardera la même représentation. Cette numérotation est non-ambiguë car, comme le graphe considéré est non orienté, on peut se restreindre à $0 \leq j < j'$ et elle avance donc bien par pas de 1.
Pour $n = 4$, les arêtes seront par exemple ordonnées comme suit :

$$[(1, 0); (2, 0); (2, 1); (3, 0); (3, 1); (3, 2)]$$

Avec cette représentation pour passer de la numérotation pour n à celle pour $n + 1$, il faut simplement rajouter à la fin les arêtes impliquant le nouveau sommet. On utilise la même représentation pour les poids :

$$W = (W_e)_{e\in E} = (W_{j,j'})_{0\leq j<j'<n} = (W_k)_{0\leq k<N}$$

On privilégiera la 3ème de ces notations. Les chaînes x sont de longueur $N = n(n-1)/2$. Pour reprendre l'exemple précédent, pour $n = 3$, le chemin $0 \to 1 \to 2 \to 0$ sera maintenant représenté par la chaîne unique :

$$x = x_{1,0}x_{2,0}x_{2,1} = x_0x_1x_2 = 111$$

Pour $n = 4$, on a les 3 chemins :

[0, 1, 2, 3]	[0, 1, 3, 2]	[0, 2, 1, 3]
101101	110011	011110

Même si le nombre de qubits est du même ordre de grandeur que précédemment ($\mathcal{O}(n^2)$), en pratique, comme le nombre de qubits disponible est très restreint, passer de 36 à 15 qubits pour $n = 6$ ou de 49 à 21 pour $n = 7$ est une bonne avancée. En effet, le simulateur de Qiskit offre 28 qubits et les machines réelles à disposition ne dépassent pas 16 qubits. Cette amélioration permet donc d'élargir le nombre de cas effectivement traitables.

Il faut naturellement trouver l'hamiltonien de coût correspondant. On décompose cet hamiltonien en plusieurs composantes comme précédemment pour pénaliser, d'une part, les chaînes ne représentant pas des chemins valides et, d'autre part, les chemins longs. Cette seconde partie est facile, on pose :

$$\mathcal{H}_1 = \sum_{k=0}^{N-1} W_k \hat{x}_k$$

Pour pénaliser les chemin invalides, on utilise l'hamiltonien suivant :

$$\mathcal{H}_2 = \sum_x (1 - \texttt{isValide}(x)) \left|x\right\rangle \left\langle x\right|$$

où $\texttt{isValide}(x)$ vaut 1 si l'état représente un chemin valide, et 0 sinon.

La construction de cet hamiltonien prend un temps exponentiel car il faut calculer $\texttt{isValide}(x)$ pour tous les x, il y en a 2^N. Cependant, étant donné un n, on a seulement besoin de calculer une seule fois cet hamiltonien. On verra dans la suite comment s'affranchir de ce calcul.

L'hamiltonien de coût total s'écrit alors :

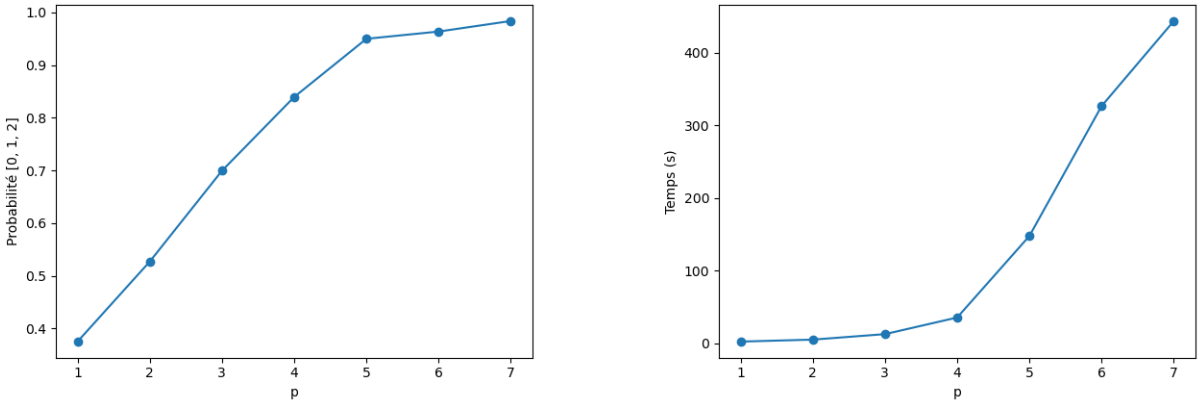
$$C = a\mathcal{H}_1 + b\mathcal{H}_2$$

avec $a \ll b$.

6.2.2 Exécution et résultats obtenus

Nous avons implémenté la méthode décrite ci-dessus avec Qiskit et testé notre programme avec différents paramètres.

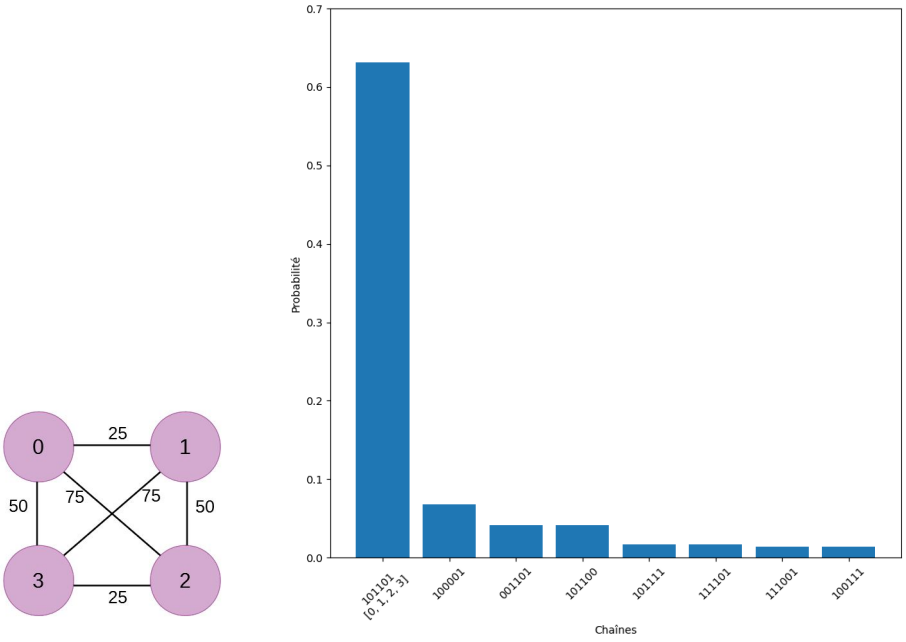
Tout d'abord, nous avons pris $n = 3$. Comme on l'a vu, dans ce cas il n'y a qu'un seul chemin possible et 7 chaînes invalides. Nous avons lancé le programme sur 50 graphes aléatoires pour différentes valeurs de p. Les poids sont tirés parmi les entiers entre 1 et 100. On a pris $a = 1$ et $b = 1000n$. Ainsi, les chemins non-valides ont des énergies au moins 10 fois plus grandes que le chemin faisable dont la longueur est majorée par $100n$. La figure 12 présente la moyenne des résultats obtenus. Ces résultats mettent en évidence que notre programme discrimine bien ces chemins invalides. Dès $p = 1$ on observe déjà de très bons résultats. Ainsi, l'algorithme décrit et notre implémentation semblent corrects. On observe que le temps d'exécution est exponentiel en p ce qui n'est pas surprenant car on réalise une recherche de minimum sur $\mathbb{R}^{2p}$.



(a) Probabilité d'obtenir le bon résultat (b) Temps d'exécution

Figure 12 – Résultats pour $n = 3$

Nous avons aussi exécuté notre programme sur quelques exemples pour $n = 4$. La figure 13 montre l'un de ces exemple avec $p = 3$. Les résultats sont aussi au rendez-vous, cependant le temps d'exécution est sensiblement plus long, environ une minute sur cet exemple ($n = 4$ et $p = 3$).



(a) Graphe test (b) Résultats

Figure 13 – Résultats pour un graphe particulier à 4 sommets et $p = 3$

Malgré tout, on observe parfois des anomalies d'exécution qui seront traitées en partie 7.4 (voir figure 18).

7 Implémentation finale

Les résultats précédents, bien qu'étant d'ores et déjà satisfaisants, mettent en lumière un point dur dans notre approche du problème. En effet, on constate sur la figure 13 que, malgré un optimum favorisé de façon non ambiguë, le second état proposé par l'algorithme (100001) ne correspond pas à un chemin viable. Il apparaît ainsi qu'une part non négligeable des états traités dans cette implémentation du QAOA le sont inutilement, et constituent donc une pollution des résultats.
Pour éviter une telle déperdition, il faut modifier le mode d'application des contraintes. Les conditions de viabilité des états doivent être imposées et ne plus simplement donner lieu à une pénalisation. Pour cela, il faut changer l'état initial, correspondant actuellement à la superposition uniforme de tous les états, et partir de la superposition uniforme des états valides. Une telle modification complexifie l'implémentation sur au moins deux plans :

— il faut réussir à construire cette superposition ;

— il faut ensuite parvenir à déterminer et à simuler un hamiltonien de conduite adapté.

Ces deux points seront traités successivement dans les parties suivantes.

7.1 Architecture générale

Le cadre théorique reste identique à celui présenté précédemment, la représentation du problème étant la même (représentation par arêtes). Comme on considérera une superposition composée uniquement des chemins viables, on n'aura plus l'usage de l'hamiltonien de pénalisation des chemins invalides $\mathcal{H}_2$. Ainsi, en gardant les mêmes notations, on a l'hamiltonien de coûts :

$$C = \sum_{k=0}^{N-1} W_k \hat{x}_k$$

Le choix de l'hamiltonien de conduite sera quant à lui détaillé plus loin. L'architecture de l'algorithme reste celle du QAOA général, avec comme modification principale l'ajout en début de circuit d'un bloc destiné à la création de la superposition des états viables.

7.2 Générer la superposition des états faisables

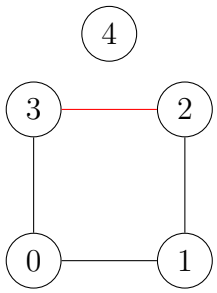
7.2.1 Présentation du circuit

Afin de réduire significativement la taille de l'espace de recherche, il est intéressant de se restreindre aux états (ou chaînes de qubits) représentant effectivement des chemins avec la représentation par arête utilisée (on garde par exemple 011110, mais pas 111111). On note $K = \frac{(n-1)!}{2}$ le nombre de chemins valides pour le TSP à n sommets. Remarquons que ce nombre est bien inférieur à 2^N états de base au total, on montre en effet que $K = o(2^N)$. On souhaite

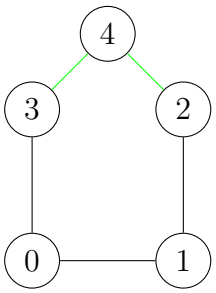
donc prendre pour état initial du QAOA non pas la superposition de tous les états comme précédemment, mais celle de tous les états représentant des chemins valides notés $(|x_n^k\rangle)_{0\leq k<K}$:

$$|\psi_n\rangle = \frac{1}{K}\sum_{k=0}^{K-1}|x_n^k\rangle$$

On peut construire cette superposition de manière itérative, à l'aide d'un circuit quantique qui prend en entrée la superposition voulue pour n sommets et crée en sortie la même superposition pour $n+1$ sommets. Cette idée est motivée par le fait que l'on peut passer ainsi d'un chemin valide de n sommets à un chemin valide de $n+1$ sommets : à partir de $|x_n^k\rangle\otimes|0\rangle^{\otimes n}$ où $|x_n^k\rangle$ représente un chemin pour n sommets et les $|0\rangle$ correspondent aux arêtes adjacentes au nouveau sommet, on retire une arête entre les sommets j et j' (on transforme un 1 en 0 dans $|x_n^k\rangle$) et on rajoute les arêtes reliant j (resp. j') à n (on transforme deux 0 en 1 dans la deuxième partie).



Chemin de taille 4.



Chemin de taille 5 généré à partir du chemin de taille 4.

A partir d'un chemin à n sommets (qui comporte donc n arêtes), on peut ainsi obtenir n chemins différents à $n+1$ sommets. En partant de tous les chemins de n sommets, on obtient une et une seule fois chaque chemin à $n+1$ sommets. En appliquant une telle transformation à $|\psi_n\rangle\otimes|0\rangle^{\otimes n}$ (en choisissant l'arête à retirer avec une probabilité uniforme), on obtient $|\psi_{n+1}\rangle$ par linéarité.

L'arbre présenté en annexe 10.1 montre les liens de parentés entre les chemins de différentes tailles. Les étiquettes sur les arêtes de l'arbre correspondent à l'arête retirée pour passer du chemin père au chemin fils.

Afin de comprendre intuitivement le fonctionnement de cet algorithme, nous allons étudier le cas non-trivial le plus simple : passer de 3 à 4 sommets, tel que représenté en annexe 10.2. Le circuit est composé de 4 registres de qubits :

— le registre x de taille $n(n+1)/2$ qui représente l'état de travail, valant initialement $|\psi_n\rangle\otimes|0...0\rangle$, et que l'on veut transformer en $|\psi_{n+1}\rangle$ à la sortie du circuit ;

— le registre a qui détermine si l'on effectue les opérations au sein d'un bloc (délimité par deux barres verticales de chaque côté) ;

— le registre b qui garde en mémoire si l'on a déjà effectué un changement ;

— le registre n qui compte le nombre de changements déjà faits.

Leur utilité apparaîtra à mesure que nous précisons l'idée de l'algorithme.

Détaillons le fonctionnement de l'algorithme. L'idée est de parcourir les $n(n-1)/2$ premiers qubits et de retirer chacune des arêtes avec une certaine probabilité. Cela se fait avec une porte Ry :

$$Ry(\theta)\,|1\rangle = \sin(\theta/2)\,|0\rangle + \cos(\theta/2)\,|1\rangle$$

On retire alors la première arête avec probabilité $1/n$ puis, dans la partie où l'arête à été conservée, on retire la deuxième avec probabilité $1/(n-1)$ conditionnellement au fait que la 1ère n'a pas été supprimée (registre b à $|0\rangle$) et ainsi de suite. On utilisera donc les angles : $\theta_k^n = 2\arcsin\sqrt{\frac{1}{n-k}}$. On a en effet :

$$Ry(\theta_k^n)\,|1\rangle = \sqrt{\frac{1}{n-k}}\,|0\rangle + \sqrt{\frac{n-k-1}{n-k}}\,|1\rangle$$

Illustrons le fonctionnement de l'algorithme pour passer de $|\psi_3\rangle$ à $|\psi_4\rangle$. On ignore ici le compteur (registre n) dont le fonctionnement sera détaillé plus loin. Les blocs sont représentés sur la figure par deux barrières (barres verticales)

— Au début, le registre x est dans l'état $|\psi_3\rangle \otimes |000\rangle = |111000\rangle$ les registres a et b sont tous les deux à $|0\rangle$.

— Au niveau de la première barrière du premier bloc, le registre a lui est passé à $|1\rangle$: il faut donc effacer l'arête représentée par x_0 avec une certaine probabilité.

— Au niveau de la deuxième barrière on a changé la première arête avec probabilité $1/3$, on est dans l'état :

$$\left(\sqrt{\frac{1}{3}}\,|011110\rangle + \sqrt{\frac{2}{3}}\,|111000\rangle\right) \otimes |10\rangle$$

— Au niveau de la barrière suivante, on modifie b et on réinitialise a, on est dans l'état :

$$\sqrt{\frac{1}{3}}\,|011110\rangle \otimes |01\rangle + \sqrt{\frac{2}{3}}\,|111000\rangle \otimes |00\rangle$$

Le registre b est à $|1\rangle$ dans la première partie de la superposition, ce qui indique qu'il ne faut plus y toucher : on a déjà retiré une arête.

— On fait la même transformation en travaillant sur le deuxième qubit. La première partie est inchangée. On a alors :

$$\sqrt{\frac{1}{3}}\,|011110\rangle \otimes |01\rangle + \sqrt{\frac{2}{3}}\left(\sqrt{\frac{1}{2}}\,|101101\rangle \otimes |01\rangle + \sqrt{\frac{1}{2}}\,|111000\rangle \otimes |00\rangle\right)$$

— Finalement, après la troisième étape on obtient bien la superposition souhaitée. Les registres a et b sont respectivement à $|0\rangle$ et $|1\rangle$ dans tous les cas. On peut donc réinitialiser b à $|0\rangle$ avec une porte X et réutiliser ces deux qubits.

Le cas général est un peu plus subtil. En effet, dans un chemin faisable, pour $n > 3$ il y aura des 0 dans le chemin (toutes les arêtes ne font pas partie du chemin). Il ne faut pas y toucher lors de l'exécution. On utilise pour cela le registre a, en plus de tester si l'on n'a pas déjà fait de changement (si b est à $|0\rangle$), on vérifie aussi qu'il y a bien une arête, c'est-à-dire que le qubit du registre x sur lequel on travaille est à $|1\rangle$. De plus on a besoin d'un compteur car on ne sait pas *a priori* combien d'arêtes (de $|1\rangle$) ont été rencontrées avant, ce qu'il faut savoir pour appliquer la bonne rotation. On explique l'exécution lorsqu'on veut passer du chemin $|101101\rangle$ à la superposition de ses 4 chemins fils. On parcourt les qubits de ce chemin.

— Le premier qubit rencontré est à $|1\rangle$ et le compteur est à 0, c'est-à-dire dans l'état $|1000\rangle$. Il faut donc supprimer cette arête avec probabilité 1/4 en appliquant une porte $R_Y(\theta_4^0)$. On a alors :

$$\sqrt{\frac{1}{4}}\,|0011011100\rangle \otimes |01\rangle + \sqrt{\frac{3}{4}}\,|1011010000\rangle \otimes |00\rangle$$

— Le deuxième qubit rencontré est à $|0\rangle$: il ne faut rien faire. Le registre a reste à $|0\rangle$ et donc aucun changement n'a lieu.

— Le troisième qubit rencontré est à $|1\rangle$ et le compteur est maintenant à 1 : $|0100\rangle$. Il faut donc supprimer cette arête avec probabilité 1/3 et donc utiliser une porte $R_Y(\theta_4^1)$. On a alors :

$$\sqrt{\frac{1}{4}}\,|0011011100\rangle \otimes |01\rangle + \sqrt{\frac{1}{4}}\,|1001010110\rangle \otimes |01\rangle + \sqrt{\frac{1}{2}}\,|1011010000\rangle \otimes |00\rangle$$

— On continue ainsi de suite sur les qubits suivants.

En pratique pour savoir quelle rotation appliquer, on teste toutes les possibilités : on applique la porte $Ry(\theta_n^k)$ pour tous les k en contrôlant par le fait que le compteur est à k c'est-à-dire en contrôlant par le k-ème qubit du compteur. La figure 14 montre un de ces blocs, celui agissant sur le 4ème qubit. Les portes **swap** au début du bloc servent à incrémenter le compteur.

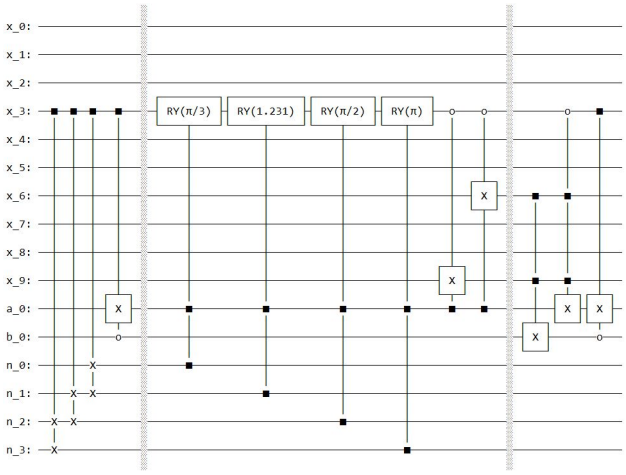


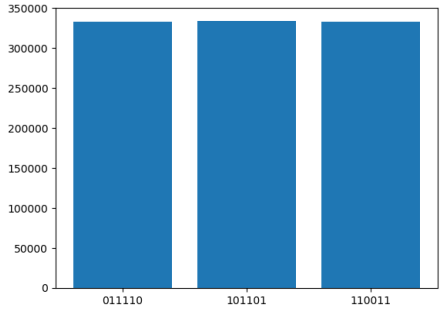
Figure 14 – Un bloc du circuit $|\psi_4\rangle \to |\psi_5\rangle$

On peut alors construire récursivement la superposition $|\psi_n\rangle$ de proche en proche, le cas de base étant le cas $n = 3$ où on a $|\psi_3\rangle = |111\rangle$ que l'on sait construire sans problème. On peut d'ailleurs réutiliser le compteur ainsi que les registres a et b pour les différentes étapes de la récursion car on sait exactement dans quel état ils sont et donc comment les réinitialiser.

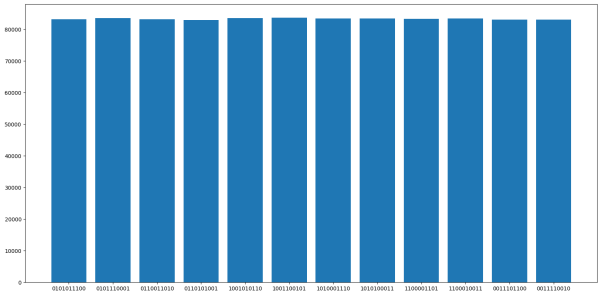
7.2.2 Validité de l'état obtenu

On peut vérifier expérimentalement la validité de l'algorithme obtenu en mesurant tous les qubits du registre contenant la représentation du chemin, et en répétant cette mesure un grand nombre de fois pour obtenir une moyenne statistique.

En appliquant notre algorithme pour $n = 4$ et $n = 5$ et en effectuant 1000000 de mesures, on obtient bien la superposition recherchée :



(a) Superposition pour $n = 4$



(b) Superposition pour $n = 5$

Figure 15 – Mesure de la superposition obtenue pour différentes valeurs de n

L'algorithme produit donc exactement le résultat voulu : l'état final est constitué de tous les états représentant des chemins valides et uniquement ceux-là (on peut le vérifier rapidement pour $n = 4$), et la superposition est uniforme.

7.2.3 Coût en qubits et en portes quantiques

Au vu de l'architecture du circuit, on établit facilement le nombre de qubits nécessaires pour générer la superposition des chemins valides à n sommets $|\psi_n\rangle$:

— $N = \frac{n(n-1)}{2}$ qubits pour représenter les chemins
— $n - 1$ qubits pour le compteur
— 2 qubits de contrôle

Il faut donc un total de $\frac{n(n+1)}{2} + 1$ qubits pour générer $|\psi_n\rangle$.

Par ailleurs, pour évaluer le coût de l'algorithme, la mesure pertinente est le nombre de portes quantiques utilisées (à rapprocher du nombre d'opérations élémentaires d'opérations pour la complexité temporelle en algorithmique classique). Au vu des différentes boucles (se référer au code fourni sur le github), le circuit utilisé fait appel à $\mathcal{O}(Nn) = \mathcal{O}(n^3)$ portes, et le circuit complet pour obtenir $|\psi_n\rangle$ contient $\mathcal{O}(n^4)$ portes quantiques.

7.3 Choix et simulation de l'hamiltonien de conduite

7.3.1 Première proposition

Conformément à la définition générale du QAOA, il faut déterminer un hamiltonien de conduite dont la superposition des états viables soit un état propre associé à la valeur propre minimale. On note cette superposition $|\psi\rangle = A\,|0...0\rangle$, où A correspond au circuit de la partie précédente, et on envisage $\hat{B}_1 = -\,|\psi\rangle\,\langle\psi|$, dont -1 est bien valeur propre minimale.
On cherche ensuite à simuler l'évolution de $|\psi\rangle$ sous l'action de $\hat{B}_1$, c'est à dire à construire un circuit quantique prenant $|\psi\rangle$ en entrée et donnant $e^{-it\hat{B}_1}\,|\psi\rangle$ en sortie.
On observe que :

$$\begin{aligned} e^{-it\hat{B}_1} &= e^{itA|0...0\rangle\langle 0...0|A^\dagger} \\ &= \sum_{k=0}^{+\infty} \frac{(it)^k}{k!} (A\,|0...0\rangle\,\langle 0...0|\,A^\dagger)^k \\ &= \sum_{k=0}^{+\infty} \frac{(it)^k}{k!} A(|0...0\rangle\,\langle 0...0|)^k A^\dagger \\ &= A e^{it|0...0\rangle\langle 0...0|} A^\dagger \end{aligned}$$

On a sous forme matricielle :

$$e^{it|0...0\rangle\langle 0...0|} = \begin{pmatrix} e^{it} & 0 & \dots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

On peut donc implémenter chaque bloc $e^{-it\hat{B}_1}$. On commence par appliquer $A^\dagger$. Ensuite on utilise un qubit auxiliaire initialement à $|0\rangle$ qu'on passe à $|1\rangle$ si tous nos qubits sont à $|0\rangle$, c'est-à-dire en lui appliquant une porte X contrôlée par le fait que tous les autres qubits soient à $|0\rangle$. Puis on applique une porte Rz avec un paramètre bien choisi. Enfin on annule toutes les opérations précédentes : on réapplique la porte X contrôlée sur le qubit auxiliaire puis on applique A.

7.3.2 Seconde proposition

On propose ici un nouvel hamiltonien de conduite. On peut construire une fonction classique, détaillée plus loin, qui induit une permutation circulaire sur les chemins en associant à un chemin un chemin suivant. Notons `Next` la porte quantique correspondante et numérotons les chemins $(|x_n^k\rangle)_{0\leq k<K}$ de telle sorte que :

$$\texttt{Next}\,|x_n^k\rangle = |x_n^{k+1}\rangle$$

où les indices sont pris modulo K.
Notons aussi $\mathcal{E}_n = \text{Vect}\,\{|x_n^k\rangle,\ 0 \leq k < K\}$ le sous espace vectoriel de $\mathbb{C}^N$ engendré par

l'ensemble des chemins faisables. On sait diagonaliser `Next` sur $\mathcal{E}$. Soit :

$$|\psi_n^j\rangle = \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} \exp\left(2i\pi \frac{jk}{K}\right) |x_n^k\rangle$$

On a bien sûr $|\psi_n\rangle = |\psi_n^0\rangle$ et $\texttt{Next}\,|\psi_n\rangle = |\psi_n\rangle$. Les $|\psi_n^j\rangle$ sont vecteurs propres de `Next` :

$$\begin{aligned}
\texttt{Next}\,|\psi_n^j\rangle &= \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} \exp\left(2i\pi \frac{jk}{K}\right) \texttt{Next}\,|x_n^k\rangle \\
&= \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} \exp\left(2i\pi \frac{jk}{K}\right) |x_n^{k+1}\rangle \\
&= \frac{1}{\sqrt{K}} \sum_{k=0}^{K-1} \exp\left(2i\pi \frac{j(k-1)}{K}\right) |x_n^k\rangle \\
&= \exp\left(-2i\pi \frac{j}{K}\right) |\psi_n^j\rangle
\end{aligned}$$

L'espace $\mathcal{E}_n$ étant de dimension K, on a bien tous les vecteurs propres. Cependant, on ne peut pas utiliser directement l'opérateur `Next` en tant qu'hamiltonien de conduite car ses valeurs propres sont complexes : le QAOA s'applique avec des opérateurs ayant des valeurs propres réelles.
Pour remédier à cela, on introduit l'opérateur $\texttt{Prev} = \texttt{Next}^\dagger$ et on pose $B_2 = -(\texttt{Prev} + \texttt{Next})$. Ainsi :

$$\begin{aligned}
B_2\,|\psi_n^j\rangle &= -\left(\exp\left(-2i\pi \frac{j}{K}\right) + \exp\left(2i\pi \frac{j}{K}\right)\right) |\psi_n^j\rangle \\
&= -2\cos\left(2\pi \frac{j}{K}\right) |\psi_n^j\rangle
\end{aligned}$$

On a donc diagonalisé B_2 sur $\mathcal{E}_n$. Notons que $|\psi_n\rangle$ est vecteur propre pour la valeur propre minimale -2. Cet opérateur convient donc pour le hamiltonien de conduite. Notons cependant que les valeurs propres de cet opérateur ne sont pas entières, on perd donc en périodicité sur la fonction objectif.

Détaillons la construction de cet hamiltonien. On explique d'abord comment réaliser la fonction `Next` de façon classique.

On se donne un chemin x de taille n et on explique comment construire le suivant. Soit j et j' les voisins du sommet $n-1$ (le dernier sommet) dans x. On commence par supprimer les arêtes $(j, n-1)$ et $(n-1, j')$ et rajouter l'arête (j, j'). On a alors transformé le chemin x en une chaîne dans laquelle $n-1$ n'a pas de voisin et induisant un chemin faisable x' sur les $n-1$ premiers sommets. Maintenant on cherche l'arête suivant (j, j') dans x' selon la numérotation présentée précédemment. On distingue alors deux cas :

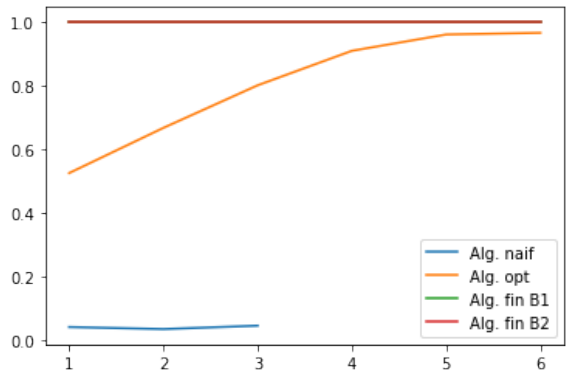
— S'il y a une telle arête (l, l'), alors on supprime cette arête et on rajoute les arêtes $(l, n-1)$ et $(n-1, l')$. On obtient alors un nouveau chemin faisable de taille n.

— S'il n'y en a pas, on calcule récursivement le chemin de taille $n-1$ suivant x' puis on cherche la première arête (l, l') de ce chemin que l'on supprime au profit des arêtes $(l, n-1)$ et $(n-1, l')$.

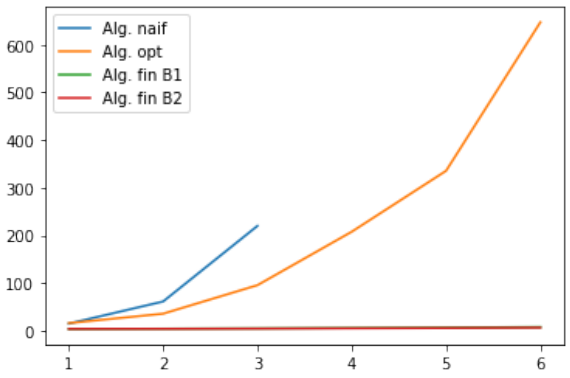
Le cas de base de la récursion est $n = 3$ où on définit le chemin suivant 111, le seul chemin faisable, comme étant lui-même. L'algorithme ainsi décrit réalise une permutation circulaire des chemins faisables. Son exécution dans le cas $n = 5$ associe à un chemin le chemin de la feuille immédiatement au dessus de lui dans l'arbre présenté en annexe 10.1.
Il faudrait alors trouver comment implémenter la porte e^{-itB_2} de façon efficace. En pratique, nous n'avons pas eu le temps de trouver cette implémentation. Nous avons donc simplement construit la matrice de taille $2^N \times 2^N$ décrivant l'opérateur B_2 à la main, en itérant la fonction classique sur tous les chemins faisables et en rentrant à la main tous les 1 dans la matrice, puis pris l'exponentielle de cette matrice, cette construction ayant bien sûr un coût prohibitif. Implémenter cette porte ne semble pas évident, cependant, la matrice B_2 n'a que deux entrées non nulles par ligne et par colonne, or dans nos recherches nous avons trouvé des articles expliquant comment approcher efficacement la porte e^{-itB_2} pour ce type de matrice en exploitant sa structure.

7.4 Analyse des résultats

On commence par comparer ces deux algorithmes aux implémentations précédentes dans les cas $n = 3$ et $n = 4$. Les figures 16 et 17 montrent les probabilités d'obtenir le meilleur chemin (a) ainsi que les temps de calcul associés (b). Les courbes rouge et verte sont superposées dans le cas de la figure 16.

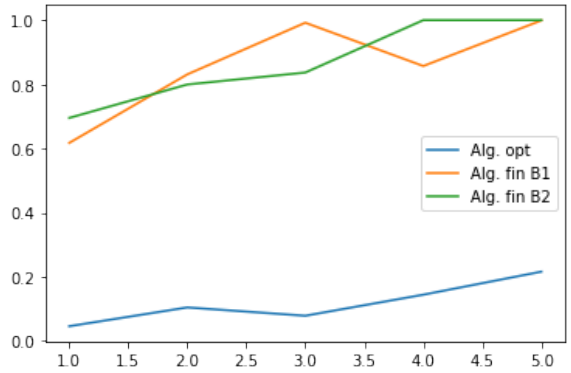


(a) Probabilité d'obtenir le bon résultat en fonction de p

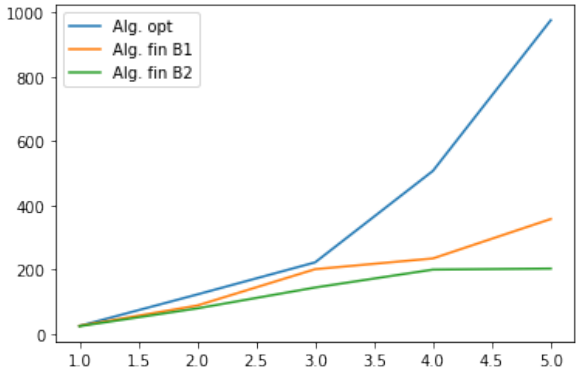


(b) Temps d'exécution (en s)

Figure 16 – Résultats pour $n = 3$



(a) Probabilité d'obtenir le bon résultat en fonction de p



(b) Temps d'exécution (en s)

Figure 17 – Résultats pour $n = 4$

Pour chaque p, on a fait une moyenne sur 10 graphes tirés au hasard. D'abord, les résultats sont bien meilleurs pour les deux implémentations finales ; en effet, comme on travaille dans un espace beaucoup plus petit que l'espace total, la probabilité se concentre sur les vrais chemins, donc il n'y a pas de perte avec les nombreux chemins infaisables. En particulier, la configuration avec 3 villes étant le cas de base pour les deux algorithmes finaux, il est logique d'obtenir systématiquement le bon résultat (unique parmi les chemins valides) en un temps de calcul constant de quelques secondes, alors que pour l'algorithme optimisé il est nécessaire d'augmenter p pour éliminer les chemins non viables. Ce dernier permet néanmoins d'illustrer l'intérêt de la représentation par arête, qui, en offrant un moins grand nombre de combinaisons non viables, réduit la déperdition de poids constatée pour l'algorithme naïf. Ce dernier est en effet totalement dépassé, tant au niveau de la qualité des résultats que de la complexité temporelle. L'augmentation rapide de cette dernière nous a conduit à ne pas le tester pour une configuration plus complexe que $n = 3$, $p = 3$.
On constate que le tracé des probabilités pour $n = 4$ est assez surprenant, puisqu'il y a deux diminutions de la qualité d'approximation alors que p croît. Afin de mieux comprendre ces résultats, on considère les données obtenues pour $p = 1$:

Alg. opt.		Alg. fin. B1		Alg. fin. B2	
T (s.)	P	T (s.)	P	T (s.)	P
34.3709	0.277077	**7.8973**	**0.333946**	**7.57**	**0.333674**
6.8329	**0.015659**	25.009	0.902605	**8.0612**	**0.333482**
6.8286	**0.015601**	**8.5429**	**0.332949**	20.103	0.903309
6.9113	**0.015406**	21.5215	0.675165	28.5279	0.912704
54.7818	0.023266	18.07833	0.653525	34.2739	0.89622
34.1578	0.01931	**11.3797**	**0.333124**	24.3212	0.790941
45.2389	0.04688	45.2435	0.857777	29.3758	0.857614
45.0665	0.013313	27.2071	0.965779	**11.494**	**0.333491**
7.7981	**0.015698**	42.5733	0.151728	30.757	0.6235
8.2709	**0.015638**	46.938	0.966056	47.1187	0.966531

On observe une corrélation entre des temps de calculs faibles et une probabilité d'environ $\frac{1}{64}$ pour l'algorithme optimisé et $\frac{1}{3}$ pour les algorithmes finaux (en gras dans le tableau ci-dessus). Ce phénomène semblerait indiquer qu'il n'y a pas eu évolution et que les états renvoyés correspondent aux états initiaux, accordant effectivement un poids uniforme à tous les états possibles, soit $\frac{1}{2^{\frac{4(4-1)}{2}}} = \frac{1}{64}$ pour l'algorithme optimisé et $\frac{1}{3}$ pour les algorithmes finaux. On peut mettre en évidence cette difficulté, par exemple pour $n = 3$, avec l'algorithme optimisé via la figure 18.

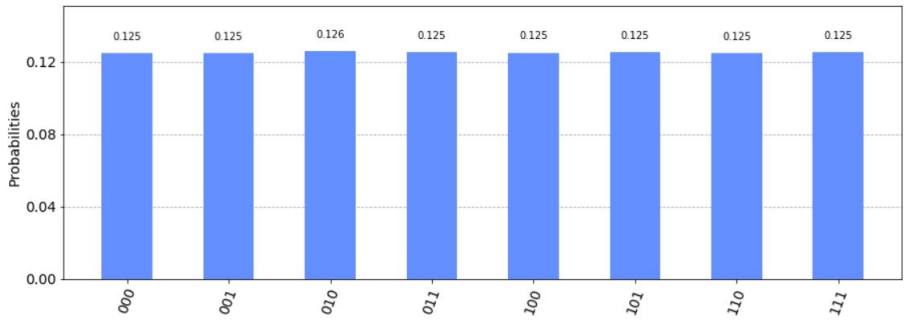


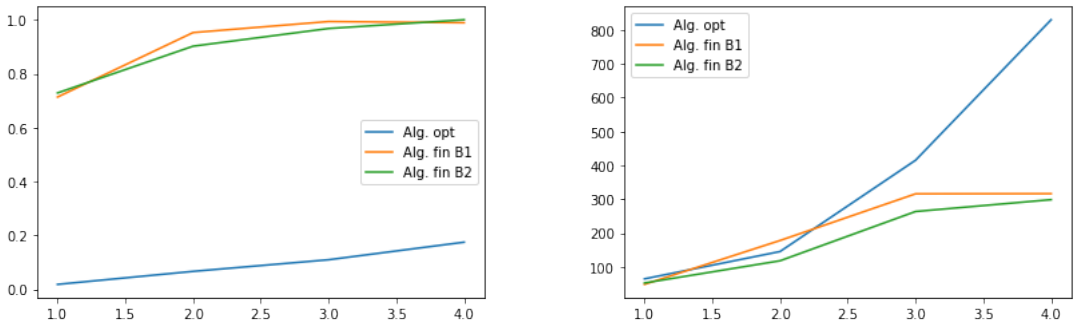
Figure 18 – Anomalie d'exécution de l'algorithme optimisé

Dans le but de stabiliser les résultats, on peut ainsi chercher à éliminer ces anomalies, en gardant à l'esprit la nécessité d'en comprendre l'origine pour pouvoir les supprimer définitivement par la suite. On propose ici deux méthodes :

— éliminer les anomalies sur la base d'un critère de temps de calcul. En effet, procéder en fonction de la probabilité ne serait pas pertinent car cela pourrait biaiser les résultats en éliminant des erreurs de calculs de l'algorithme. Or ici, l'objectif est de ne retirer que les exécutions où le calcul n'a pas eu lieu, pas celle où il a donné un mauvais résultat ;

— augmenter le panel de test, en passant à une moyenne sur 100 graphes, évitant ainsi l'interférence avec les résultats potentiellement causée par la première méthode.

Via la première approche, on obtient la figure 19 pour $n = 4$:



(a) Probabilité d'obtenir le bon résultat

(b) Temps d'exécution

Figure 19 – Résultats pour $n = 4$ avec critère temporel de réussite

Cette première méthode confirme les résultats des graphiques précédents en termes de qualité d'approximation et de complexité, tout en évitant les anomalies de décroissance en p de la qualité d'approximation.
Néanmoins, elle ne permet pas de distinguer clairement les drivers B_1 et B_2. Pour cela, on utilise la seconde approche avec laquelle on obtient la figure 20, toujours pour $n = 4$:

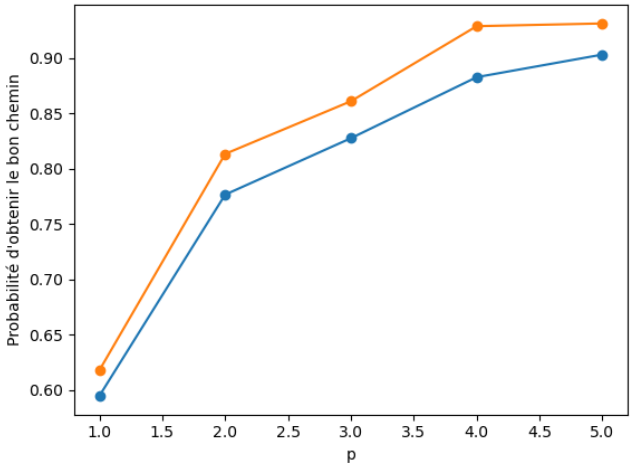


Figure 20 – Probabilité d'obtenir le bon chemin avec les drivers B_1 (en bleu) et B_2 (en orange)

On remarque que le driver B_2 semble systématiquement meilleur que B_1. Ceci s'explique peut-être par le fait que B_2 est diagonalisable dans le sous-espace des chemins faisables.

Pour revenir sur les anomalies mises en évidence, l'apparente non réalisation du calcul semblerait indiquer une erreur au niveau de la phase d'optimisation des paramètres du QAOA. Nous n'avons malheureusement pas eu le temps de résoudre cette difficulté, qui reste donc une piste d'amélioration de nos implémentations.

8 Déroulement du projet et organisation du travail

Le fonctionnement de groupe et l'organisation du travail ont, dans l'ensemble, été conformes à ce qui avait été prévu, bien que la répartition des tâches envisagée à l'origine ait légèrement évolué après la phase III.

Afin d'assurer l'efficacité du travail collectif ainsi qu'une compréhension théorique de chaque sujet abordé par tous les membres du groupe, le déroulement du projet a en effet été divisé en cinq phases - deux purement consacrées à un travail bibliographique (I et II), une à la découverte et à l'étude du QAOA (III), puis deux plus orientées vers la mise en oeuvre concrète des sujets étudiés (IV et V) :

- **Phase I** : Pour s'assurer une connaissance précise des algorithmes classiques de résolution du TSP, nous nous sommes répartis les différentes familles d'algorithmes existantes, chaque membre du groupe préparant une présentation de celle qu'il avait choisie. Nous avons ainsi acquis une compréhension collective approfondie du fonctionnement de chacune de ces méthodes.
- **Phase II** : Dans une approche similaire à celle de la phase précédente, nous avons fonctionné sur la base d'exposés afin d'approfondir notre connaissance des algorithmes quantiques canoniques.
- **Phase III** : Étude collective de l'algorithme QAOA et début de l'implémentation.
- **Phase IV - V** : Réflexions collectives et individuelles pour la construction de l'algorithme. Fondé sur le schéma global du QAOA, ce dernier a notamment évolué au gré des choix de représentations et d'hamiltoniens de conduite présentés dans ce rapport. En dehors de cette partie centrale qu'a été l'évolution de l'implémentation, cette phase a également permis de renforcer notre compréhension théorique du QAOA, d'analyser les résultats obtenus et de travailler sur la complexité des algorithmes développés.

Voué avant tout à structurer le projet, ce découpage n'a pas toujours été respecté d'un point de vue chronologique, avec notamment une implémentation précoce et un retour important sur les aspects théoriques du QAOA en deuxième partie de projet.
Ainsi, pour ce qui est de l'organisation, nous nous sommes peu ou prou conformés à celle prévue dans la proposition détaillée, c'est-à-dire :

- un **fonctionnement séquentiel** pour les phases I, II et III, notamment par la présentation commune d'exposés de nos recherches bibliographiques respectives.
- un **fonctionnement en parallèle** pour les phases IV et V.

Les phases I, II et III se sont déroulées avec succès, tous les membres du groupe ayant pu prendre connaissance de l'état d'avancée des approches classiques, des éléments constitutifs de l'informatique quantique, des différentes approches quantiques du TSP, et enfin des raisons du choix du QAOA.

Le passage de la phase III à la phase IV s'est quant à lui effectué de façon efficace quoique perfectible. En effet, le principe de fonctionnement en parallèle, bien qu'offrant flexibilité et efficacité, se couple d'une difficulté que, dans un premier temps, nous n'avions pas pris en compte. Le fait que chacun n'ait à avancer que sur un aspect donné du projet, couplé à l'importance

cruciale de disposer d'un algorithme fonctionnel à analyser, a conduit à une production rapide d'un code. Il fut alors tentant d'y apporter toutes les optimisations envisageables, internes comme externes, de façon à accroître au plus vite ses capacités. Ce faisant, la dynamique du groupe s'en est retrouvée impactée. En effet, la nécessaire appropriation du code par tous a mécaniquement conduit à des séances uniquement consacrées à la présentation de la dernière version de ce code. Bien que bénéfique sur plusieurs aspects, cette évolution accélérée de notre algorithme a ainsi présenté certaines difficultés :

— Une polarisation entre premier de cordée, rédacteur du nouveau code, et lecteurs de ce dernier.

— Une compréhension théorique insuffisamment développée pour saisir l'ensemble des implications des modifications apportées ou du nouveau paradigme choisi.

Nous avons donc procédé à un ajustement de notre fonctionnement interne, de façon à pouvoir profiter de toutes les compétences disponibles, en évitant la paralysie associée à un fonctionnement inadéquat au format de ce projet. Cet ajustement a été caractérisé par une répartition rééquilibrée de l'**investissement collectif**, en consacrant notamment une plus grande attention aux **aspects théoriques** des concepts étudiés. Rétrospectivement, cette modification de notre fonctionnement interne s'est révélée adaptée aux phases IV et V, en permettant un renforcement de notre compréhension théorique du QAOA ainsi que des évolutions importantes de notre implémentation.

9 Conclusion

En conclusion, ce PSC a été l'occasion de découvrir une approche innovante et prometteuse pour la résolution d'un problème fondamental en informatique, le TSP.
Sur le plan technique, on peut résumer notre démarche comme suit :

— rechercher la représentation du problème minimisant le nombre de qubits utilisés ;

— implémenter l'algorithme de façon à utiliser le moins grand nombre de portes possible, afin de limiter sa profondeur et donc la décohérence de l'état de travail ;

— améliorer notre compréhension des résultats en couplant le développement du code avec une étude théorique.

Les optimisations réalisées ont, quant à elles, permis :

— de réduire le nombre de qubits utilisés de n^2 à $\frac{n(n-1)}{2}$;

— d'augmenter la vitesse et la précision des calculs par l'emploi de fonctions avancées de Qiskit.

— de supprimer des résultats les parcours non-viables, permettant une amélioration significative de la précision de l'approximation tout en limitant la complexité temporelle.

Nos résultats laissent néanmoins entrevoir d'autres points à améliorer, notamment en termes de choix d'hamiltonien de conduite et de calcul des paramètres γ et β optimaux. Ainsi, on pourrait à l'avenir envisager les améliorations suivantes :

— développer une compréhension du lien entre QAOA et théorie du contrôle optimal, tel que décrit dans la référence [Yan+17] ;

— mener des analyses plus poussées sur l'évolution des populations lors de l'application de nos implémentations du QAOA, à l'image des résultats de [Zho+20] présentés en partie 5.4 ;

— poursuivre l'exploitation de références telles que [Zho+20] ou [SL19] qui proposent des méthodes permettant d'améliorer le calcul de γ et β, par exemple en exploitant une régularité dans la structure des paramètres optimaux pour les estimer de façon précise au rang $p+1$, à partir de leur connaissance au rang p dans le cas de [Zho+20] ;

— résoudre le problème des anomalies d'exécution ;

— simuler quantiquement l'évolution suivant l'hamiltonien de conduite B_2.

On pourrait ainsi espérer augmenter de façon significative les p atteignables et la qualité d'approximation de l'algorithme.
Enfin, ce projet aura également été l'occasion d'améliorer notre fonctionnement collectif sur un projet de longue durée, permettant, par là même, un développement humain tout autant que technique.

Références

[Far+00] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. arXiv : quant-ph/0001106.

[AR06] Andris Ambainis et Oded Regev. *An Elementary Proof of the Quantum Adiabatic Theorem*. 2006. arXiv : quant-ph/0411152.

[GP06] Gregory Gutin et Abraham P Punnen. *The traveling salesman problem and its variations*. T. 12. Springer Science & Business Media, 2006.

[Cro+14] Elizabeth Crosson et al. *Different Strategies for Optimization Using the Quantum Adiabatic Algorithm*. 2014. arXiv : 1401.7320 [quant-ph].

[FGG14] Edward Farhi, Jeffrey Goldstone et Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv : 1411.4028 [quant-ph].

[Yan+17] Zhi-Cheng Yang et al. « Optimizing Variational Quantum Algorithms Using Pontryagin's Minimum Principle ». In : *Physical Review X* 7.2 (mai 2017). issn : 2160-3308. doi : 10.1103/physrevx.7.021027. url : http://dx.doi.org/10.1103/PhysRevX.7.021027.

[Sun+18] Yin Sun et al. *Adiabatic Quantum Simulation Using Trotterization*. 2018. arXiv : 1805.11568 [quant-ph].

[HD19] Daniel Henry et Sofia Josefina Lago Dudas. « Solving the Traveling Salesman Problem Using QAOA ». In : (2019).

[RMS19] Matthew Radzihovsky, Joey Murphy et Mason Swofford. « A QAOA solution to the traveling salesman problem using pyQuil ». In : (2019).

[SL19] Michael Streif et Martin Leib. *Training the Quantum Approximate Optimization Algorithm without access to a Quantum Processing Unit*. 2019. arXiv : 1908.08862 [quant-ph].

[VBB19] Guillaume Verdon, Michael Broughton et Jacob Biamonte. *A quantum algorithm to train neural networks using low-depth circuits*. 2019. arXiv : 1712.05304 [quant-ph].

[Rua+20] Yue Ruan et al. « The Quantum Approximate Algorithm for Solving Traveling Salesman Problem ». In : *Computers, Materials and Continua* 63.3 (2020), p. 1237-1247.

[Zho+20] Leo Zhou et al. « Quantum Approximate Optimization Algorithm : Performance, Mechanism, and Implementation on Near-Term Devices ». In : *Physical Review X* 10.2 (juin 2020). issn : 2160-3308. doi : 10.1103/physrevx.10.021067. url : http://dx.doi.org/10.1103/PhysRevX.10.021067.

[Bou] Olivier Bournez. *Fondement de l'informatique. Logique, modèle et calculs. Chapitre 13 : Quelques problèmes NP-Complets*. url : https://www.enseignement.polytechnique.fr/informatique/INF412/uploads/Main/chap13-goodINF412.pdf.

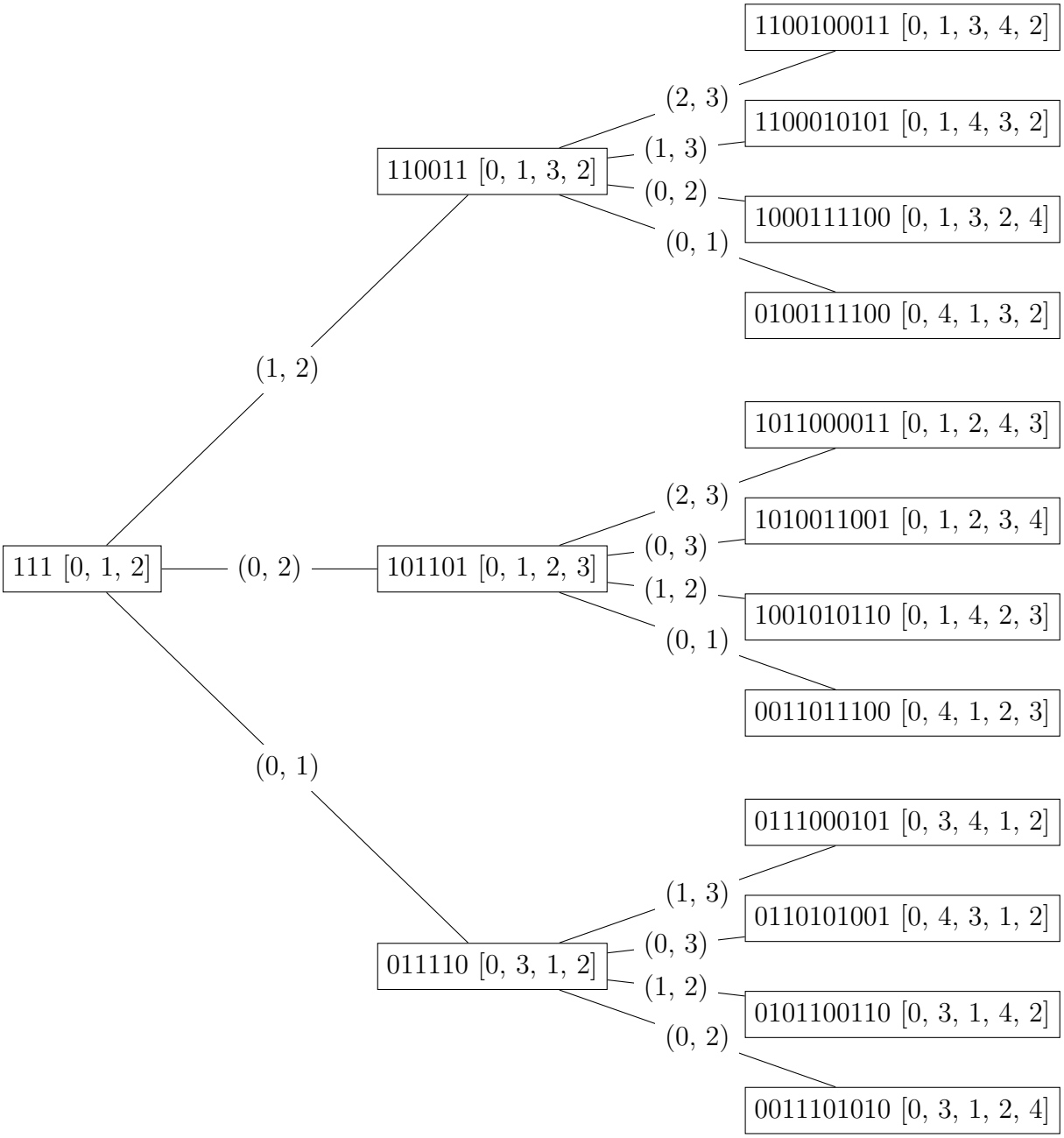
[Com] The Jupyter Book Community. *Quantum Phase Estimation*. URL : https://qiskit.org/textbook/ch-algorithms/quantum-phase-estimation.html.

[Hab] Howard Haber. *The time evolution operator as a time-ordered exponential*. URL : http://scipp.ucsc.edu/~haber/ph215/TimeOrderedExp.pdf.

[Jof] Manuel Joffre. *Physique Quantique Avancée*. URL : https://moodle.polytechnique.fr/pluginfile.php/204338/mod_resource/content/8/phy430.pdf.

10 Annexes

10.1 Illustration de l'algorithme de superposition



10.2 Circuit permettant le passage $|\psi_3\rangle \to |\psi_4\rangle$