

# Mémoire projet de fin d'étude

Pour l'obtention du diplôme d'ingénieur en informatique

Option : Systèmes Informatiques (SIQ)

## Thème

Résolution de problèmes d'ordonnancement flow shop en  
utilisant les algorithmes quantiques

Réalisé par :

Saiem Malak

Encadré par :

Pr. Fatima Benbouzid-Si Tayeb  
(LMCS-ESI)

Dr. Taha Arbaoui (LOSI-UTT)

Dr. Faicel Hnaïen (LOSI-UTT)

---

# DEDICACES

## À DIEU TOUT PUISSANT

À ma source d'inspiration, chère mère et chère tante, les deux femmes de ma vie qui ont été toujours une école de patience, d'amour et de confiance, qui m'ont toujours supportée, ces deux femmes qui représentent une source d'inspiration pour moi, elles m'ont chaleureusement accompagné et encouragé tout au long de mon parcours.

À mon père ci-chaleureux, ma petite princesse POPINA et mes frères Mahdi et Mamoune qui m'ont toujours soutenue et encouragée.

À la mémoire de mes grand-parents. Puisse Dieu vous avoir en sa sainte miséricorde et que ce travail soit une prière pour vos âmes.

À mes chères amies Fatima, Inès et Saadia. et à ma deuxième famille GDG Algiers.

# Remerciements

Ce travail n'aurait pu aboutir sans l'aide et le soutien que m'ont témoigné les nombreuses personnes côtoyées durant cette année. Qu'il me soit permis de leur témoigner ici ma profonde reconnaissance et les remercier pour tout ce qu'ils m'ont apporté.

Tout d'abord, je dois commencer par remercier Allah, le tout-puissant, de m'avoir donné la santé, la volonté, le courage et la patience pour mener à terme ma formation et pouvoir réaliser ce travail de recherche.

J'exprime mes plus sincères et chaleureux remerciements à mon encadrante Mme. Si Tayeb pour ses orientations, sa disponibilité et ses conseils judicieux.

Je tiens aussi à exprimer ma profonde gratitude à mes superviseuses, Monsieur Taha Arbaoui et Monsieur Faicel Hnaïen qui m'ont donné la chance de travailler sur ce formidable sujet de recherche, pour leur patience, leur disponibilité et leurs conseils qui m'ont aidé à alimenter ma réflexion.

À ma famille, je vous remercie de m'avoir inculquée le goût du savoir et de l'ambition, de m'avoir toujours supportée.

J'adresse aussi mes sincères remerciements à tous mes enseignants de l'ESI, leurs conseils et leurs critiques m'ont guidé pendant tout mon cursus et m'ont permis d'avoir la connaissance nécessaire pour pouvoir intégrer ce projet.

## Résumé

L'informatique quantique a connu ces dernières années un progrès très important en termes théorique ainsi que de la disponibilité des dispositifs quantiques. Les algorithmes quantiques offrent des améliorations considérables dans les applications d'optimisation et d'apprentissage automatique, notamment dans les domaines de la finance, de l'énergie, de l'automobile et des sciences environnementales. En effet, les problèmes d'optimisation combinatoire (POC) souvent extrêmement complexes, que l'informatique actuelle n'arrive pas à résoudre efficacement, voire pas du tout, représentent un vaste champ d'application pour l'informatique quantique.

Principalement, ces méthodes se distinguent en deux types, le premier qui combine les principes de calcul quantique avec les méthodes de résolution des POCs classique afin d'améliorer ces derniers. Le deuxième type représente des nouvelles approches quantiques proposées dans le cadre de la résolution des POCs tel que les méthodes basées sur l'algorithme de Grover et les algorithmes variationnels quantiques.

Parmi les problèmes d'optimisation combinatoire les plus connus dans les systèmes de manufacture et de production, nous trouvons les problèmes d'ordonnancements de type Flow Shop. en effet, l'importance de ces problèmes se justifie par la large gamme des interprétations que les ressources et les tâches peuvent prendre.

Les travaux menés dans le cadre de ce projet de fin d'étude consiste à la résolution d'un problème d'ordonnancement de type Flow Shop avec les approches quantique.

**Mots Clés :** Informatique quantique, Calcul quantique, Algorithmes variationnels, Algorithme de Grover, Les problèmes d'optimisation combinatoires, Les problèmes d'ordonnancement, FlowShop

## Abstract

The extensively covered quantum mechanism is advancing all the time. The robust processing capability and high parallelism of the quantum mechanism give the quantum field considerable vitality and a wide range of application scenarios. This discipline has made significant strides in recent years, both in terms of theory and the accessibility of quantum devices. The fact that many quantum algorithms outperform classical ones has encouraged researchers to apply this technology in other fields like chemistry, finance, and artificial intelligence.

Quantum combinatorial optimization is a novel area of study that aims to create new methods for solving combinatorial optimization problems based on the principles of quantum computation. These techniques can primarily be divided into two categories, the first of which combines quantum computation principles with traditional combinatorial optimization problem (COP) solution techniques. The second class consists of novel quantum methods that have been put forth to address combinatorial optimization issues (COPs).

One of the most well known combinatorial optimization problems is scheduling problems that aim to find an optimal schedule, reduces cost, and takes into consideration the different resource constraints. However, it has been found that it is very difficult to produce constant-quality orders. This is because most heuristic strategies are difficult to evaluate, and finding the solutions that correspond to the most practical schedule necessitates costly and time-consuming enumeration operations.

In this context, We focus on solving the Flowshop scheduling problem to minimize the makespan ( $F_m|perm|C_{max}$ ) using an exact and a heuristic quantum approaches.

**Keywords :** Combinatorial Optimization, Quantum Computing, FlowShop, Variational methods, Grover's Algorithm, Scheduling problems

# Table des matières

<b>Acronymes</b>	<b>x</b>
<b>1 L'informatique quantique</b>	<b>4</b>
1.1 Introduction . . . . .	4
1.2 Principes de la mécanique quantique . . . . .	5
1.3 L'information quantique . . . . .	6
1.3.1 Le bit quantique . . . . .	7
1.3.2 Système quantique à plusieurs qubits . . . . .	8
1.3.3 Manipulation d'un système quantique . . . . .	9
1.3.4 Le calcul quantique adiabatique . . . . .	13
1.4 Les algorithmes quantiques . . . . .	13
1.4.1 les algorithmes variationnels . . . . .	14
1.4.2 Le recuit quantique . . . . .	15
1.5 Les ordinateurs quantiques . . . . .	16
1.5.1 Les simulateurs quantiques . . . . .	16
1.5.2 Les ordinateurs quantiques à portes logiques . . . . .	17
1.5.3 Langage de programmation et Kits de développement logiciels . . .	17
1.5.4 Simulation des circuits quantique sur les ordinateurs classiques . .	18
1.6 Enjeux et défis liés à la réalisation d'un système quantique . . . . .	19
1.7 Conclusion . . . . .	21
<b>2 Optimisation combinatoire quantique</b>	<b>22</b>
2.1 Introduction . . . . .	22
2.2 Les problèmes d'optimisation combinatoire - Définition et Complexité . . .	23
2.3 Résolution des POCs sur un ordinateur quantique . . . . .	25
2.4 Classification des méthodes quantique pour la résolution des POCs . . . .	28
2.4.1 Méthodes de résolution des POCs sur un ordinateur quantique . . .	29
2.4.2 Méthodes hybrides de résolution des POCs . . . . .	32
2.4.3 Méthodes de résolution des POCs inspirés quantiques sur un ordi- nateur classique . . . . .	36

2.5	Les méthodes de résolution quantique proposés pour la résolution des problèmes d'ordonnancements . . . . .	38
2.5.1	Éléments de base d'un problème d'ordonnement . . . . .	38
2.5.2	Notation des problèmes d'ordonnement . . . . .	40
2.5.3	Synthèse des travaux basés quantique proposés dans la littérature pour la résolution des problèmes d'ordonnement . . . . .	41
2.6	Synthèse des méthodes de résolution quantique proposés pour la résolution des problèmes d'optimisation combinatoire . . . . .	43
2.7	Conclusion . . . . .	47
<b>3</b>	<b>Approche quantique basée sur l'algorithme de Grover</b>	<b>48</b>
3.1	Introduction . . . . .	48
3.2	Contexte de l'étude - Le Flow-Shop de permutation . . . . .	48
3.3	Architecture générale de l'approche . . . . .	50
3.4	Phase 1 : Encodage de la solution . . . . .	51
3.5	Phase 2 : Construction du circuit de Grover . . . . .	54
3.5.1	Étape 1 : L'initialisation . . . . .	55
3.5.2	Étape 2 : L'opérateur oracle . . . . .	56
3.5.3	Étape 3 : L'opérateur de diffusion . . . . .	64
3.6	Phase 3 : Optimisation globale avec l'algorithme de Grover . . . . .	65
3.6.1	Algorithme basé sur l'approche GAS . . . . .	65
3.6.2	Algorithme basé sur la Recherche binaire . . . . .	68
3.7	Conclusion . . . . .	69
<b>4</b>	<b>Approches quantiques basée sur les algorithmes variationnels</b>	<b>70</b>
4.1	Introduction . . . . .	70
4.2	Modélisation mathématique . . . . .	71
4.2.1	Un modèle QUBO basé sur le modèle QUBO du TSP . . . . .	72
4.2.2	Un modèle QUBO basé sur le modèle Position index . . . . .	76
4.3	L'hamiltonien . . . . .	78
4.4	Algorithme variationnel quantique . . . . .	78
4.4.1	Variational Quantum Eigensolver . . . . .	80
4.4.2	Quantum Approximate Optimisation Algorithm . . . . .	83
4.4.3	QAOA-Swap . . . . .	84
4.5	Conclusion . . . . .	87
<b>5</b>	<b>Implémentation, Tests et Analyse des performances</b>	<b>88</b>
5.1	Outils . . . . .	88
5.1.1	Langage de programmation . . . . .	88
5.1.2	Librairies . . . . .	88

5.1.3	Plateforme de développement / IDE . . . . .	90
5.1.4	Plateforme d'exécution . . . . .	90
5.1.5	Simulateur quantique . . . . .	91
5.2	Implémentation des approches proposés . . . . .	91
5.3	Les benchmarks de tests . . . . .	92
5.4	Analyse des performances de Grover-FSP . . . . .	92
5.4.1	Génération du circuit de Grover . . . . .	92
5.4.2	Tests du circuit du Grover . . . . .	94
5.4.3	Analyse des performances du circuit de Grover . . . . .	98
5.5	Analyse des performances de VQE-FSP . . . . .	100
5.5.1	Comparaison entre les différents modèles QUBO proposés . . . . .	100
5.5.2	Comparaison entre les différentes méthodes implémentés . . . . .	105
5.6	Conclusion . . . . .	107



# Table des figures

1.1	Représentation de la sphère de Bloch d'un qubit (tirée du livre " Quantum Computation and Quantum Information" <b>Isaac Chuang (2002)</b> ) . . . . .	8
1.2	L'opérateur Toffoli <b>Isaac Chuang (2002)</b> . . . . .	11
1.3	Représentation de la mesure dans un circuit quantique . . . . .	12
1.4	Un simple circuit quantique comportant deux opérateurs, l'opérateur Hadamard et le CNOT (tirée du livre " Quantum Computation and Quantum Information" <b>Isaac Chuang (2002)</b> ) . . . . .	13
1.5	Schéma général présentant le processus d'exécution des algorithmes quantique hybride <b>Cerezo et al. (2021)</b> . . . . .	15
1.6	Changement d'énergies et distribution de probabilités lors de l'exécution d'un recuit quantique, tiré de <b>Leonidas (2021)</b> . . . . .	15
2.1	Les liens possibles entre le calcul quantique et la théorie de la complexité <b>Vert (2021)</b> . . . . .	25
2.2	Reformulation d'un POC . . . . .	25
2.3	Classification des méthodes de résolution de POCs . . . . .	29
2.4	L'inversion autour de la moyenne tirée de <b>Durr &amp; Hoyer (1996)</b> . . . . .	30
2.5	Représentation géométrique de l'algorithme de recherche de Grover dans un espace à deux dimensions. (a) Représentation de l'état initial dans l'espace bidimensionnel des états cible ( $ ci\rangle$ ) et non cible ( $ c \perp\rangle$ ). (b) Action des opérateurs D et V sur l'état initial <b>Zahedinejad &amp; Zaribafiyani (2017)</b> . . . . .	31
2.6	Schéma général présentant le processus d'exécution des algorithmes quantique hybride VQE <b>Qiskit &amp; Pattanayak (n.d.)</b> . . . . .	33
2.7	Le processus d'exécution de l'algorithme de QAOA. <b>Rieffel et al. (2020)</b> . . . . .	34
2.8	La qualité des résultats de test de QAOA pour la résolution des problèmes de maxcut avec 10 noeuds, tiré de <b>Cook et al. (2020)</b> . . . . .	35
2.9	présente le timeline d'apparition des premières méthodes de résolution des POCs quantique . . . . .	43
3.1	Schéma général présentant l'approche Grover-FSP . . . . .	51
3.2	Initialisation . . . . .	56

3.3	Oracle . . . . .	56
3.4	Inversion de la phase . . . . .	57
3.5	L'initialisation des registres $C_{ij}$ . . . . .	58
3.6	Initialisation de $ C_{ij}\rangle$ . . . . .	59
3.7	$ C_{ij}\rangle$ . . . . .	60
3.8	Initialisation . . . . .	63
3.9	Vérification si deux positions comportent le même job . . . . .	64
3.10	L'opérateur de la diffusion . . . . .	64
3.11	Décomposition du problème de Flow Shop . . . . .	65
3.12	Organigramme de la méthode GAS . . . . .	66
4.1	Architecture générale de l'approche . . . . .	71
4.2	Architecture générale de VQE . . . . .	80
4.3	Le circuit paramétré <b>Two Local</b> . . . . .	82
4.4	Architecture générale de QAOA-FSP . . . . .	83
4.5	Le swap mixer pour une instance de trois jobs . . . . .	86
5.1	Diagramme de classe . . . . .	91
5.2	Générateur du circuit de Grover . . . . .	92
5.3	Circuit de Grover généré . . . . .	93
5.4	Oracle généré pour deux tâches . . . . .	94
5.5	Test 1 avec le nombre de rotations égale à 1 . . . . .	95
5.6	Test 1 avec le nombre de rotations égale à 2 . . . . .	95
5.7	Test 1 avec le nombre de rotations égale à 3 . . . . .	95
5.8	Test 2 avec le nombre de rotations égale à 1 . . . . .	96
5.9	Test 2 avec le nombre de rotations égale à 2 . . . . .	96
5.10	Test 2 avec le nombre de rotations égale à 3 . . . . .	97
5.11	Test 3 avec le nombre de rotations égale à 1 . . . . .	97
5.12	Test 3 avec le nombre de rotations égale à 2 . . . . .	98
5.13	Test 3 avec le nombre de rotations égale à 3 . . . . .	98
5.14	Nombre de qubits en fonction de nombre de tâches . . . . .	99
5.15	Résultats obtenus après l'exécution de l'approche VQE avec les différents modèles QUBO 1-6 . . . . .	102
5.16	Résultats obtenus après l'exécution de l'approche QAOA swap avec les différents modèles QUBO 1-6 . . . . .	103
5.17	Résultats obtenus après l'exécution de l'approche QAOA avec les différents modèles QUBO 1-6 . . . . .	104
5.18	Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de temps d'exécution . . . . .	105

5.19 Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de la qualité de la solution obtenu . . . . .	106
--	-----

# Liste des tableaux

1.1	Les opérateurs quantiques Umeano (2021) . . . . .	12
1.2	Les simulateurs des circuits quantiques . . . . .	19
2.1	Les contraintes classiques les plus connus et leurs pénalités équivalentes. . .	26
2.2	Récapitulatif des travaux sur la résolution des problèmes d’optimisation combinatoire avec l’algorithme de Grover . . . . .	44
2.3	Récapitulatif des travaux sur la résolution des problèmes d’optimisation combinatoire avec des méthodes Hybrides . . . . .	45
3.1	Les temps d’exécution de 4 jobs sur les deux machines . . . . .	54
4.1	Approximation de FSP vers TSP . . . . .	73
5.1	Résultats de l’exécution des algorithmes GAS et BSG . . . . .	100
5.2	Résultats obtenus avec l’approche VQE avec les différentes approches 1-6 .	101
5.3	Résultats obtenus après l’exécution de l’approche QAOA avec les différentes approches 1-6 . . . . .	103
5.4	Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de la qualité de la solution obtenu . . . . .	106

# Liste des Algorithmes

1	Algorithmes génétique inspiré quantique . . . . .	37
2	Algorithme évolutionnaire inspiré quantique . . . . .	38
3	Algorithme de génération de l'opérateur $init  C_{ij}\rangle$ . . . . .	60
4	Pseudo Algorithme de génération de l'opérateur $\Omega$ . . . . .	61
5	Algorithme de génération du circuit pour calculer le makespan . . . . .	62
6	Algorithme de recherche adaptative basée sur Grover . . . . .	67
7	Algorithme de recherche binaire basée sur Grover . . . . .	68
8	Pseudo-Algorithme VQE . . . . .	82
9	Pseudo-Algorithme QAOA . . . . .	84
10	Pseudo-Algorithme QAOA Swap . . . . .	86

# Acronymes

**BSG** Binary Search Grover. ix

**CPBO** Constrained Polynomial Binary Optimization. ix

**CVaR** Conditional Value-atRisk. ix

**F-VQE** Filtering Variational Quantum Eigensolver. ix

**GAS** Grover Adaptive Search. ix

**Layer VQE** Layer Variational Quantum Eigensolver. ix

**NISQ** Noisy Intermediate Scale Quantum. ix

**PFSP** Permutation Flow Shop. ix

**POC** Problème d'optimisation combinatoire. ix

**PQC** Parameterized quantum circuit. ix

**QAOA** Quantum Approximative Optimization Algorithm. ix

**QC** Quantum Computer. ix

**QEC** Quantum error correction. ix

**QGA** Quantum Gentic Algorithm. ix

**QUBO** Quadratic unconstrained BinaryOptimisation. ix

**RSA** Rivest-Shamir-Adleman. ix

**TSP** Travelling salesman Problem. ix

**VQA** Variational Quantum Algorithms. ix

**VQE** Variational Quantum Eigenseolver. ix

# Introduction Générale

Aujourd'hui, les entreprises évoluent dans un monde dans lequel le marché est plus concurrentiel et le consommateur est plus conscient et exigeant en matière de qualité, de coût et de délai. L'objectif principal de ces derniers est de pouvoir offrir à ses clients des solutions satisfaisant leurs besoins dans les meilleurs délais tout en minimisant les coûts de production. En effet, l'optimisation de ces systèmes représente un enjeu majeur pour les décideurs qui cherchent à exploiter la puissance des technologies disponibles aujourd'hui pour pouvoir satisfaire ses clientèles.

L'ordonnancement des ateliers représente un des axes d'optimisation le plus important pour ces systèmes, il consiste principalement à ordonnancer les opérations dans les processus de production de manière optimale ou quasi optimale, d'une façon à minimiser les coûts et en prenant en considération les différentes contraintes de disponibilités. Toutefois, il s'est avéré extrêmement difficile de générer des ordonnancements de qualité constante et ceci revient au fait que trouver les solutions qui représentent des ordonnancements optimaux impliquent des procédures d'énumération coûteuses et peu pratiques, tandis que la performance de la plupart des techniques heuristiques est difficile à évaluer.

Ces dernières années, un nouvel axe de recherche est apparu, il s'agit de l'axe qui consiste à exploiter la puissance de l'informatique quantique pour pouvoir résoudre les problèmes d'optimisation combinatoire. Cette voie représente un sujet de recherche incontournable et prometteur qui promet de révolutionner un grand nombre de domaines. En effet, l'intérêt dans ce domaine a grandi depuis la proposition des algorithmes de Grover et Shor qui ont montré des performances significatives par rapport à leurs homologues classiques **Grover (1996); Isaac Chuang (2002)**. En outre, le modèle de calcul quantique semble être le seul modèle de calcul disponible aujourd'hui capable de surmonter les limites de la loi de Moore et d'explorer les nouvelles frontières dont le calcul classique n'est pas encore au point pour les résoudre efficacement **National Academies of Sciences & Medicine (2019); Isaac Chuang (2002)**. Ce domaine basé sur les concepts du monde de l'infiniment petit représente un sujet de recherche incontournable qui promet de révolutionner un grand nombre de domaines tel que l'apprentissage automatique, la cryptographie, la chimie et la finance. Une autre voie d'application qui s'avère très prometteuse est celle qui essaye de développer de nouvelles approches qui s'appuient sur les principes de la théorie quantique pour traiter plus efficacement les problèmes d'optimisa-

tion combinatoire (POCs).

Principalement, nous pouvons distinguer deux branches pour la résolution des problèmes d'optimisation combinatoire quantique. La première combine les concepts du calcul quantique avec les algorithmes de résolution de POCs classiques

(Ross, 2019; Narayanan & Moore, 1996; Li *et al.*, 2020; Han & Kim, 2002). Ces méthodes ont montré leurs performances pour la résolution des POCs. La deuxième branche utilise des algorithmes quantiques pour la résolution des POCs. Cette branche représente un axe de recherche très actif qui combine des algorithmes basés sur différents types de calcul quantique tel que les méthodes basées sur le calcul quantique adiabatique comme le recuit quantique Vert (2021) et les méthodes basées sur le modèle des portes logiques Grover (1996); Liu *et al.* (2022).

Notre projet de fin d'étude se situe à la frontière de l'optimisation combinatoire quantique et de l'informatique quantique et a pour objectif de donner un aperçu sur l'application du calcul quantique dans la résolution des problèmes de l'optimisation combinatoire et de proposer des approches quantiques pour la résolution d'un problème d'ordonnement de type Flow-Shop.

Le mémoire de notre projet de fin d'étude est divisé en deux parties. La première partie représente une étude bibliographique sur l'optimisation combinatoire quantique. Cette partie comporte deux chapitres :

- **Le chapitre 1** présente les fondements de l'informatique quantique. Dans un premier lieu, nous allons commencer par présenter les principes de la physique quantique ainsi que les concepts de la théorie de l'information quantique. Par la suite, nous allons introduire les modèles de calcul quantique existants ainsi que les algorithmes quantiques. La dernière partie de ce chapitre consiste à présenter les défis et les enjeux liés à la conception des systèmes quantiques.
- **Le chapitre 2** présente les notions de l'optimisation combinatoire quantique ainsi qu'une classification des différentes approches quantiques existantes. Ensuite, nous allons présenter comment écrire un POCs sous forme d'un modèle mathématique quadratique sans contraintes, pour pouvoir le résoudre avec certaines méthodes quantiques.

La deuxième partie de notre mémoire représente un cas d'application des méthodes quantiques hybrides et des méthodes basées sur l'algorithme de Grover pour la résolution du problème du Flow-Shop de permutation (PFSP). Cette partie est organisée en trois chapitres.

- **Le chapitre 3** intitulé "Approche quantique basées sur l'algorithme de Grover" présente une approche pour la résolution du PFSP. Nous présenterons l'architecture générale de l'approche puis nous détaillerons les différentes phases de réalisation du circuit quantique.



- **Le chapitre 4** intitulé "Approches quantiques basée sur les algorithmes variationnels" présente les différentes approches variationnelles hybrides proposées pour la résolution du PFSP. Nous commencerons par introduire le modèle mathématique proposé pour la modélisation linéaire du PFSP puis nous introduirons les différentes approches quantique hybrides proposées.
- **Le chapitre 5** introduit les choix des techniques d'implémentation (Langage de programmation, Bibliothèque, simulateur quantique,...), puis nous expliquerons comment nous avons généré les benchmarks de tests. Nous présenterons aussi les différents tests effectués ainsi qu'une analyse des performances des méthodes proposées et les limites de ces derniers.

# Chapitre 1

## L'informatique quantique

*“The history of the universe, is in effect, a huge ongoing quantum computation. The universe is a quantum computer.”*  
–Seth Lloyd

### 1.1 Introduction

En 1982, Richard Feynman a proposé pour la première fois de construire une machine quantique basée sur les principes de la physique de l'infiniment petit pour simuler les problèmes réels avec des systèmes quantiques que les ordinateurs classiques ne sont pas capables de simuler **Feynman (1986)**. Depuis, le domaine n'a cessé de se développer. En effet, le potentiel de ce paradigme de calcul a commencé à s'accroître depuis l'apparition de l'algorithme de Shor qui a permis de résoudre le problème de factorisation des nombres entiers d'une façon exponentiellement meilleure que les algorithmes classiques existants et qui peut être utilisé pour casser le système de cryptage asymétrique RSA **Shor (1999)** et de l'algorithme de Grover qui permet d'obtenir une amélioration quadratique pour la résolution des problèmes de recherche **Grover (1996)**.

Par ailleurs, il est important de souligner que le progrès des ordinateurs classiques édicté la « loi de Moore » a postulé que le nombre de composants dans chaque circuit intégré doublerait tous les deux ans tout en gardant le même coût de fabrication **Moore (1975)**. Il est clair que cette tendance exponentielle de progrès ne peut être maintenue indéfiniment, et même s'il y a beaucoup de débats sur le moment exact où cette mise à l'échelle cessera, les signes de sa fin ont commencé au cours de cette décennie **Theis & Wong (2017)**. De ce fait, le modèle de calcul quantique semble pouvoir être une deuxième révolution permettant de surmonter cette contrainte.

Dans ce chapitre, nous allons présenter les concepts du calcul quantique. Dans un premier lieu, nous introduisons les concepts fondamentaux de la mécanique quantique ainsi que les principes de la théorie de l'information quantique. En outre, nous présentons le côté pratique de l'informatique quantique ainsi que les défis et les enjeux liés à ce domaine.

## 1.2 Principes de la mécanique quantique

La mécanique quantique a la particularité d'être à la fois la plus aboutie et la plus mystérieuse de nos théories scientifiques. Elle a été développée par à-coups sur une période remarquable allant de 1900 aux années 1920. Dans les décennies qui ont suivi les années 1920, les physiciens ont eu beaucoup de succès dans l'application de la mécanique quantique pour comprendre les particules et les forces fondamentales de la nature, qui a abouti à l'élaboration du modèle standard de la physique des particules **Isaac Chuang (2002)**. Au cours de la même période, les physiciens ont eu un succès tout aussi grand dans l'application de la mécanique quantique pour comprendre une gamme étonnante de phénomènes de notre monde allant des polymères aux semi-conducteurs et des superfluides aux supraconducteurs. Aujourd'hui, la mécanique quantique est considérée comme étant le modèle physique le plus complet qui décrit avec une grande précision la nature à l'échelle microscopique, les interactions entre atome, particules élémentaires et éventuellement l'univers entier **Vert (2021)**.

Dans ce domaine, on appelle souvent ces atomes et petites particules par entité ou objets quantiques. Ils possèdent plusieurs propriétés non intuitives, qui ont donné naissance à une nouvelle façon de modéliser et de traiter l'information. Les propriétés les plus importantes qui sont exploitées dans le calcul quantique sont : (i) l'intrication quantique, (ii) la superposition, (iii) l'équation d'évolution et (iv) l'effondrement de la fonction d'onde **National Academies of Sciences & Medicine (2019)**. Ces propriétés, étant primordiales pour la compréhension du paradigme quantique, seront décrites brièvement dans ce qui suit.

### (i) Intrication quantique

L'intrication quantique (appelée aussi l'enchevêtrement) entre deux ou plusieurs particules s'exprime comme l'impossibilité de décrire l'état d'une particule indépendamment de l'état d'autres particules, même si ces états sont séparés par une grande distance **National Academies of Sciences & Medicine (2019)**. Les objets quantiques intriqués partagent la même fonction d'onde, ce qui lui accorde un état quantique corrélé sans être pour autant identique. Ce principe stipule qu'on ne peut pas dissocier les particules en enchevêtrement jusqu'à la mesure. La mesure d'une particule parmi les autres fait perdre aux autres leur état de superposition aussi **Vert (2021)**. L'intrication quantique représente un rôle clé dans de nombreuses applications du calcul quantique et de l'information quantique telles que la téléportation quantique, les algorithmes quantiques rapides et la correction d'erreur quantique.

- (ii) **La superposition** La superposition représente un des principes fondateurs de la physique quantique. Elle stipule qu'un objet quantique peut se trouver dans un ou plusieurs états à la fois et toute opération effectuée sur cet objet s'effectue en

parallèle sur tous les états superposés. Prenons comme exemple un photon qui peut prendre deux directions de polarisation différentes (Nord et Est). Avec le principe de superposition, le photon peut être polarisé dans une direction Nord, dans une direction Est ou les deux en même temps. Nous pouvons exprimer l'état quantique comme étant une combinaison linéaire à coefficient complexe d'états observables **Montanaro (2016)**. Ces coefficients représentent les amplitudes correspondantes à chaque état et ils ont une norme égale à un. La superposition d'un état quantique  $|\psi\rangle$  en notation de Dirac s'exprime comme suit :

$$|\psi\rangle = a_0|\psi_0\rangle + a_1|\psi_1\rangle + a_2|\psi_2\rangle + ..... + a_{N-1}|\psi_{N-1}\rangle + a_N|\psi_N\rangle$$

L'état quantique  $|\psi\rangle$  représente l'état global du système. Les  $a_i$  représentent les amplitudes et les  $|\psi_i\rangle$  représentent l'ensemble des états observables **Vert (2021)**. L'importance de ce principe réside dans le fait qu'il permet de manipuler plusieurs états à la fois avec un seul système quantique **National Academies of Sciences & Medicine (2019)**.

(iii) **L'interférence**

Ce concept est décrit dans l'expérience du dispositif à double fente de Young, où la lumière passe par deux dispositifs à double fente et produit des régions sombres et claires en raison des interférences constructives et destructives respectivement. **Umeano (2021)**. Ce principe représente le fait que lorsque nous disposons de deux fonctions d'onde en superposition associées à deux amplitudes complexes. Leurs amplitudes peuvent s'accumuler (interférences constructives) comme ils peuvent s'annihiler (interférences destructives).

(iv) **L'effondrement de la fonction d'onde**

En mécanique quantique, le concept de l'effondrement de la fonction d'onde indique que lorsqu'un système quantique est mesuré, il perd sa propriété de superposition et se réduit à un seul état. Cela signifie qu'à partir d'un système quantique décrit par une superposition d'états possibles, on ne peut extraire qu'une seule valeur après mesure **National Academies of Sciences & Medicine (2019)**.

## 1.3 L'information quantique

Entre les années 1970 et 1980, certains chercheurs se sont posés la question fondamentale de savoir si l'informatique et la théorie de l'information pouvaient être appliquées à l'étude des systèmes quantiques **Isaac Chuang (2002)**. Au lieu de considérer les systèmes quantiques comme des phénomènes à expliquer tels qu'on les trouve dans la nature, ils les ont considérés comme des systèmes pouvant être conçus. Le monde quantique n'est plus simplement considéré comme présenté, mais de nouvelles perspectives ont été défi-

nis et les fondements de la mécanique quantique et de la théorie de l'information sont apparus.

Cette section est considérée comme une introduction à la théorie de l'information quantique. Dans un premier lieu, nous allons définir la notion de qubit, de systèmes quantique à plusieurs qubits. Puis, nous introduirons les différents aspects du calculs quantiques.

### 1.3.1 Le bit quantique

Le bit quantique ou souvent appelé le qubit représente la notion fondamentale de la théorie de l'information quantique. Tout comme dans un ordinateur classique, on utilise le bit comme unité de mesure de l'information classique, dans un ordinateur quantique, on utilise des qubit comme unité de base pour décrire l'état d'un système quantique **Jaffali (2020)**. Contrairement au bit qui peut prendre la valeur 0 ou la valeur 1, le qubit peut être, selon la notation de Dirac  $\langle ket|bra \rangle$ , dans l'état 0 dénoté  $|0\rangle$ ; l'état 1 dénoté  $|1\rangle$ ; mais aussi dans une superposition des deux états  $|0\rangle$  et  $|1\rangle$  exprimé comme une combinaison linéaire des deux états. En effet, le qubit possède toutes les propriétés quantiques citées auparavant **Isaac Chuang (2002)**.

Un qubit en superposition de deux états de base  $|0\rangle$  et  $|1\rangle$ , est représenté par l'état  $|\psi\rangle$  comme suit :

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

Où  $\alpha$  et  $\beta$  sont des nombres complexe représentant l'amplitude de probabilité des états  $|0\rangle$  et  $|1\rangle$ , et respectant la contrainte de normalisation.

$$|\alpha|^2 + |\beta|^2 = 1$$

Le qubit est défini comme un vecteur dans l'espace de Hilbert de dimension 2, où les vecteurs  $|0\rangle$  et  $|1\rangle$  sont les vecteurs qui forment la base orthonormée **Feynman et al. (2018)**. Ces vecteurs sont représentés par :

$$|0\rangle \sim \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle \sim \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

L'état  $|\psi\rangle$  peut aussi être représenté comme ci-dessous :

$$|\psi\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \in H = \mathbb{C}^2$$

Une autre reformulation géométrique du qubit est la suivante :

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

$\theta, \varphi$  représente des nombres réels tels que  $0 \leq \theta \leq \pi$  et  $0 \leq \varphi \leq 2\pi$ . Cette représentation permet de représenter un qubit comme un point sur une sphère (appelé la sphère de Bloch) pour bien visualiser l'état du système ainsi que les résultats d'application des opérateurs logique comme présenté dans 1.1.

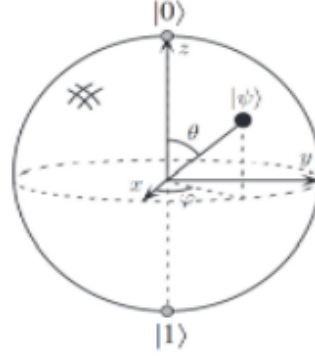


FIGURE 1.1 – Représentation de la sphère de Bloch d'un qubit (tirée du livre " Quantum Computation and Quantum Information" **Isaac Chuang (2002)**)

### 1.3.2 Système quantique à plusieurs qubits

Il est primordial de pouvoir modéliser un système avec plusieurs qubits et de pouvoir décrire l'état global de ce système et aussi l'état de chaque particule. Considérant l'état  $|AB\rangle$  représentant l'état d'un système quantique à deux particules représentées par deux qubits A et B. Le qubit A peut être en superposition des deux états  $|0_A\rangle$  et  $|1_A\rangle$ . Le qubit A est un vecteur de l'espace de Hilbert  $H_A$  de dimension 2. De même pour la particule B qui peut être exprimé comme un vecteur de l'espace de Hilbert  $H_B$  de dimension 2 **Jaffali (2020)**. L'état  $|AB\rangle$  du système global peut être aussi représenté comme vecteur dans un espace de Hilbert de dimension 4, défini comme un produit tensoriel des espaces de Hilbert des particules A et B. On a alors  $H_{AB} = H_A \otimes H_B$ .

Comme le système contient deux particules A et B, l'état du système doit être exprimé comme la combinaison de tous les états possibles  $|0_A\rangle|0_B\rangle, |0_A\rangle|1_B\rangle, |1_A\rangle|0_B\rangle, |1_A\rangle|1_B\rangle$  **Jaffali (2020)**. Afin de calculer l'état global du système, nous appliquerons le produit tensoriel entre A et B comme suit :

$$|\psi_{AB}\rangle = |\psi_A\rangle + |\psi_B\rangle$$

$$|\psi_{AB}\rangle = \begin{pmatrix} \alpha_A \\ \beta_A \end{pmatrix} \otimes \begin{pmatrix} \alpha_B \\ \beta_B \end{pmatrix}$$

$$|\psi_{AB}\rangle = \begin{pmatrix} \alpha_A * \begin{pmatrix} \alpha_B \\ \beta_B \end{pmatrix} \\ \beta_A * \begin{pmatrix} \alpha_B \\ \beta_B \end{pmatrix} \end{pmatrix}$$

En appliquant le produit tensoriel entre les deux vecteurs A et B on obtiendra un vecteur dans un espace de Hilbert de dimension 4 avec les vecteurs de base  $|00\rangle$ ,  $|01\rangle$ ,  $|10\rangle$ ,  $|11\rangle$ .

$$|\psi_{AB}\rangle = \begin{pmatrix} \alpha_A * \alpha_B \\ \alpha_A * \beta_B \\ \beta_A * \alpha_B \\ \beta_A * \beta_B \end{pmatrix} \quad (1.1)$$

D'après l'équation (1.1) les probabilités d'obtenir les états  $|00\rangle$ , le  $|01\rangle$ , le  $|10\rangle$  et le  $|11\rangle$  sont  $(\alpha_A\alpha_B)^2$ ,  $(\alpha_A\beta_B)^2$ ,  $(\beta_A\alpha_B)^2$  et  $(\beta_A\beta_B)^2$  respectivement. Par conséquent, l'état  $|AB\rangle$  peut être représenté comme suit :

$$|\psi_{AB}\rangle = \alpha_A\alpha_B |00\rangle + \alpha_A\beta_B |10\rangle + \beta_A\alpha_B |01\rangle + \beta_A\beta_B |11\rangle$$

En utilisant le même principe, on peut générer un état pour décrire un système avec n qubit. L'état  $|\psi_N\rangle$  qui décrit un système avec n-qubit s'exprime comme la superposition de toutes les valeurs de 0 à  $N = 2^n$  :

$$|\psi_N\rangle = \alpha_0 |000...0\rangle + \alpha_1 |000...1\rangle + \alpha_2 |000...10\rangle + ..... + \alpha_N |111..11\rangle$$

Dans ce cas, l'état  $|\psi_N\rangle$  représente un vecteur dans un espace de Hilbert de dimension n, définit comme un produit tensoriel des espaces de Hilbert  $H_n = H_1 \otimes H_2 \otimes H_3.... \otimes H_n$  **Isaac Chuang (2002)**.

### 1.3.3 Manipulation d'un système quantique

De la même manière que pour les ordinateurs classiques, les ordinateurs quantiques utilisent différents modèles de calcul et chaque modèle possède son propre principe du calcul. Principalement, nous pouvons distinguer les modèles de calcul quantique suivant : le circuit quantique, le modèle quantique à sens unique, le modèle quantique topologique et le modèle de calcul quantique adiabatique **Umeano (2021)**. Les circuits quantiques sont les modèles les plus utilisés et peuvent être définis comme une séquence de transformation unitaire décrite par une matrice U avec des entrées complexes. Il représente l'équivalent complexe d'une transformation orthogonal pour les espaces vectoriel réel **Jaffali (2020)**. Ces matrices, appelées portes logiques ou les opérateurs, permettent de passer d'un état quantique vers un autre en appliquant des modifications sur les amplitudes des états. Ils ont comme propriété la réversibilité, c.-à-d. en appliquant le même opérateur deux fois en peut revenir à l'état initial. Nous pouvons distinguer deux types d'opérateurs quantiques : (1) les opérateurs qui permettent de changer l'état d'un seul qubit qui sont représentés par des matrices unitaires carrées et (2) les opérateurs qui permettent de changer l'état d'un système à plusieurs qubits qui agissent sur des états avec n qubits. Ces derniers sont représentés par des matrices unitaires  $2^n \times 2^n$  **Isaac Chuang (2002)**.

### 1.3.3.1 Les opérateurs quantiques sur un seul qubit

Ces opérateurs agissent sur un seul qubit seulement. Ils sont exprimés comme une matrice carrée d'ordre 2 unitaire **Isaac Chuang (2002)**. Les opérateurs quantiques les plus connus sont :

1. L'opérateur Pauli X qui permet d'inverser les amplitudes des qubits. il est considéré comme L'équivalent quantique du NOT booléen.

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

2. L'opérateur Pauli Z qui permet de modifier la phase de  $\varphi$ .

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle$$

3. L'opérateur Pauli Y qui permet de changer la phase et l'amplitude, comme suit :

$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle$$

4. L'opérateur identité I qui possède la même fonction de l'identité. En l'appliquant sur un qubit, on garde le même état.

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

5. L'opérateur Hadamard H permettant de créer une superposition uniforme à partir de l'état  $|0\rangle$ .

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

En appliquant H sur l'état  $|0\rangle$ , on obtient un nouvel état de superposition ou la probabilité d'être en  $|0\rangle$  où  $|1\rangle$  est la même  $\frac{1}{2}$ .




### 1.3.3.2 Les opérateurs quantiques sur un système à plusieurs qubits

On peut aussi utiliser des opérateurs pour changer l'état global d'un système à  $n$  qubits. Ces opérateurs sont exprimés comme une matrice carrée d'ordre  $2^n$ . Avec ces opérateurs, on peut appliquer des opérations complexes sur les états des systèmes et manipuler ces derniers **Isaac Chuang (2002)**.

1. L'opérateur C-NOT appelé aussi controlled-NOT est un opérateur qui agit sur deux qubits. Cet opérateur remplit la fonction de "if", si le bit de qubit de contrôle est à 1, on change la valeur de qubit cible sinon elle reste inchangé **Isaac Chuang (2002)**. La matrice correspondante à cet opérateur est :

$$U_{C-NOT}|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle$$

2. L'opérateur Toffoli est un opérateur qui agit sur 3 qubits. Il est similaire à l'opérateur C-NOT mais avec deux qubits de contrôle. La figure 1.2 présente cet opérateur ainsi que la matrice correspondante.



$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

FIGURE 1.2 – L'opérateur Toffoli **Isaac Chuang (2002)**

Le tableau 1.1 présente un récapitulatif des opérateurs quantiques les plus utilisés.


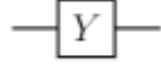
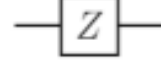

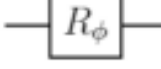
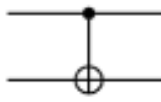
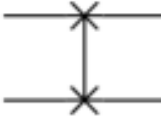
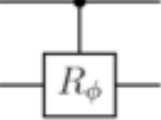
Name	Circuit	Matrix	Operation
Pauli-X (Quantum NOT)		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\sigma_x 0\rangle =  1\rangle$ $\sigma_x 1\rangle =  0\rangle$
Pauli-Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\sigma_y 0\rangle = i 1\rangle$ $\sigma_y 1\rangle = -i 0\rangle$
Pauli-Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\sigma_z 0\rangle =  0\rangle$ $\sigma_z 1\rangle = - 1\rangle$
Hadamard		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$H 0\rangle = \frac{ 0\rangle+ 1\rangle}{\sqrt{2}} ( +\rangle)$ $H 1\rangle = \frac{ 0\rangle- 1\rangle}{\sqrt{2}} ( -\rangle)$
Phase shift		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$	$R_\phi 0\rangle =  0\rangle$ $R_\phi 1\rangle = e^{i\phi}  1\rangle$
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	$U_{CN} 00\rangle =  00\rangle$ $U_{CN} 01\rangle =  01\rangle$ $U_{CN} 10\rangle =  11\rangle$ $U_{CN} 11\rangle =  10\rangle$
SWAP		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$SWAP 00\rangle =  00\rangle$ $SWAP 01\rangle =  10\rangle$ $SWAP 10\rangle =  01\rangle$ $SWAP 11\rangle =  11\rangle$
Controlled-phase		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}$	$U_{C_\phi} 00\rangle =  00\rangle$ $U_{C_\phi} 01\rangle =  01\rangle$ $U_{C_\phi} 10\rangle =  10\rangle$ $U_{C_\phi} 11\rangle = e^{i\phi}  11\rangle$

TABLE 1.1 – Les opérateurs quantiques Umeano (2021)

Généralement, le calcul quantique se termine par une mesure afin de réduire l'état quantique dans l'un des états de base. Dans le circuit quantique, cette opération est représentée comme le montre la figure 1.3.

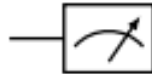


FIGURE 1.3 – Représentation de la mesure dans un circuit quantique

### 1.3.4 Le calcul quantique adiabatique

Au lieu d'utiliser une suite de portes quantique en vue de réaliser les calculs, le modèle quantique adiabatique (AQC) utilise le principe de l'évolution d'un Hamiltonien initial vers un Hamiltonien final, dont l'état fondamental représente la solution **Umeano (2021)**. Comme son nom l'indique, ce modèle est basé sur le théorème adiabatique qui stipule *"Pour un système donné, si ce système évolue assez lentement, alors il restera toujours dans le même état d'énergie propre"* Max Born and Vladimir Folk 1928.

Pour comprendre ce théorème, nous supposons que nous avons un Hamiltonien  $H_P$  dont l'état fondamental encode la solution de notre problème et un Hamiltonien initial  $H_0$  dont l'état fondamental est facile à préparer. Si nous préparons un système quantique pour qu'il soit dans l'état fondamental de  $H_0$ , ensuite, nous le faisons évoluer à travers le temps vers un Hamiltonien  $H$  tel que :

$$H(t) = (1 - \frac{t}{T})H_0 + \frac{t}{T}H_P$$

alors notre système restera dans son état fondamental à tout moment, ce qui signifie que pour  $t = T$  notre système sera dans l'état fondamental de  $H_P$  qui est notre solution.

## 1.4 Les algorithmes quantiques

En calcul quantique, un problème est représenté comme une superposition des états quantiques possibles. À chaque état est associée une probabilité, et afin de pouvoir obtenir l'état désiré avec la probabilité la plus élevée, l'algorithme quantique manipule les amplitudes des états en appliquant une série d'opérations quantiques. Chaque opération s'exprime comme une multiplication par une matrice unitaire dans le cas de modèle de circuit quantique. Tandis que sur le modèle AQC les opérations quantiques s'exprime comme l'évolution d'un Hamiltonien comme décrit dans la section précédente **Isaac Chuang (2002)**.

Dans le modèle à portes quantiques, la séquence des opérations quantiques appliquée est décrite sous la forme d'un circuit, comme illustré dans la figure 1.4.

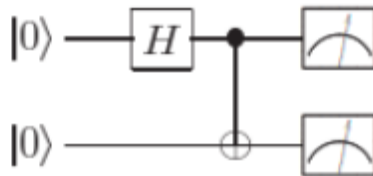


FIGURE 1.4 – Un simple circuit quantique comportant deux opérateurs, l'opérateur Hadamard et le CNOT (tirée du livre " Quantum Computation and Quantum Information" **Isaac Chuang (2002)**))

Nous pouvons distinguer quatre catégories d'algorithmes quantiques **Vert (2021)** :

1. Les algorithmes basés sur la transformée de Fourier quantique (QFT) qui utilise l'algorithme de Shor pour résoudre des problèmes comme les problèmes de factorisation et de logarithme discret. Ces algorithmes permettent de casser un grand nombre de cryptosystèmes, y compris le cryptosystème RSA **Isaac Chuang (2002)**.
2. Les algorithmes de recherche comme l'algorithme de Grover et de Deutsch-Jozsa.
3. Les algorithmes d'optimisation qui cherchent à trouver un minimum (ou maximum) dans un système complexe **Vert (2021)** comme le recuit quantique et les algorithmes variationnels.
4. Les algorithmes de simulations quantiques permettant de simuler les systèmes quantiques (les interactions entre des atomes, les interactions dans des structures moléculaires ...) **Vert (2021)**.

Il existe nombreux types d'algorithmes qui ont été conçus pour la résolution des problèmes de différents domaines tel que la chimie, la cryptographie, l'optimisation combinatoire et l'apprentissage automatique.

### 1.4.1 les algorithmes variationnels

Ces méthodes sont des méthodes approchées hybrides qui exploitent l'ordinateur quantique pour obtenir une fonction d'onde paramétrée qui permet de générer un état quantique  $\psi(\theta)$  et l'ordinateur classique pour maximiser/minimiser la valeur attendue de l'Hamiltonien du problème **Cerezo et al. (2021)** comme le montre la figure 3.1. En général, nous créons un circuit quantique paramétré qui dépend d'un certain angle  $\theta$  et ensuite l'ordinateur classique va essayer de trouver les meilleurs paramètres qui permettent de minimiser /maximiser l'énergie du l'Hamiltonien  $C(\theta) = E(\theta) = \langle \psi(\theta) | H | \psi(\theta) \rangle$  **Leonidas (2021)**. Ce type d'algorithme était suggéré pour contourner le problème de la lenteur du recuit quantique et pour pouvoir implémenter l'Hamiltonien du problème à résoudre dans le matériel disponible **Leonidas (2021)**.

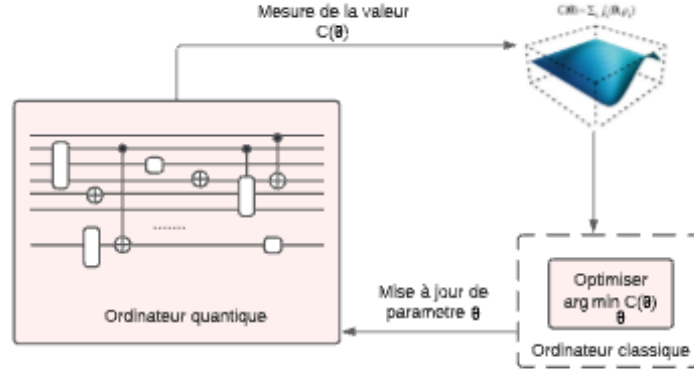


FIGURE 1.5 – Schéma général présentant le processus d'exécution des algorithmes quantique hybride **Cerezo et al. (2021)**

### 1.4.2 Le recuit quantique

Le recuit quantique (En anglais Quantum annealing QA) est une méthode approchée permettant de trouver une solution approximative à un problème combinatoire à valeur fini basée sur le principe du théorème quantique adiabatique **Vert (2021)**. Elle représente une version quantique du recuit simulé classique qui au lieu de manipuler des fluctuations thermiques classiques, utilise des fluctuations thermiques quantiques pour rendre l'exploitation de l'espace de configuration plus rapide **Vert (2021)**. L'idée de cette méthode est d'initialiser le système par un état fondamental d'un Hamiltonien facile à réaliser et simple puis faire varier lentement le premier Hamiltonien avec l'Hamiltonien correspondant à la fonction de coût du problème étudiée pour pouvoir trouver l'état correspondant à la solution du problème **Leonidas (2021)**. En 2000, Farhi et al. ont proposé un modèle adiabatique pour la résolution des problèmes d'optimisation combinatoire **Farhi et al. (2000)**. Cette méthode ne peut pas être exécutée sur un ordinateur quantique à porte logique **Umeano (2021)**. Actuellement, c'est l'entreprise D-Wave qui a développé un solveur de recuit quantique pour la résolution des problèmes d'optimisation combinatoire exprimé sous la forme QUBO **Leonidas (2021)**.

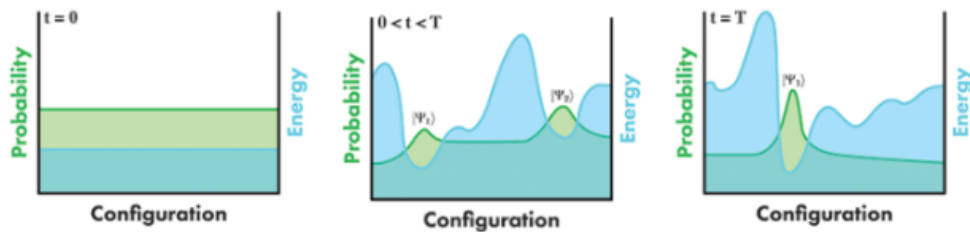


FIGURE 1.6 – Changement d'énergies et distribution de probabilités lors de l'exécution d'un recuit quantique, tiré de **Leonidas (2021)**

La figure 1.6 présente comment l'énergie et la distribution de probabilité changent lors de l'exécution du recuit quantique. Nous voyons que pour  $t = 0$ , l'énergie et la probabilité sont distribuées de manière égale. Pour  $0 \leq t \leq T$  nous voyons que la distribution de l'énergie commence à augmenter, tandis que la probabilité pour ces minimums est plus grande que les autres valeurs de l'énergie. Enfin, pour  $t = T$  nous voyons que l'énergie a une valeur minimale pour laquelle la probabilité est la plus grande **Leonidas (2021)**.

## 1.5 Les ordinateurs quantiques

En plus de la création de matériel quantique avec un taux d'erreur faible, un ordinateur quantique fonctionnel nécessite aussi des outils logiciels comme les langages de programmations, les compilateurs, les simulateurs ainsi que les solutions cloud et les kits de développement. Dans ce qui suit, nous allons présenter les différents types d'ordinateurs quantiques ainsi que les outils de programmation quantique, les enjeux et les défis liés à la conception et la réalisation des systèmes quantiques. Essentiellement, on peut distinguer deux types d'ordinateurs quantiques, les ordinateurs quantiques qui font la simulation quantique et les ordinateurs à porte logique qui sont les plus proches des ordinateurs classiques **Gondran & Gondran (2020)**.

### 1.5.1 Les simulateurs quantiques

Cette catégorie comporte les ordinateurs quantiques qui font la simulation quantique. On peut distinguer deux types de simulateurs : (1) le premier représente les ordinateurs quantiques analogiques connus par les ordinateurs à la Feynman. Ces ordinateurs ont été construits à la base pour la reproduction des phénomènes complexes quantiques difficiles à simuler sur des supercalculateurs **Gondran & Gondran (2020)**. Ils sont semblables aux ordinateurs classiques analogiques qui existaient avant le numérique et pour l'instant ces ordinateurs n'apportent pas de performances. La deuxième catégorie représente les ordinateurs quantiques adiabatiques comme ceux de D-wave. Ce type d'ordinateurs est adaptée seulement pour l'exécution d'un type d'algorithmes quantiques appelés algorithmes à recuit quantique **Vert (2021)**. Ils utilisent une technique similaire au recuit simulé où le système est initialisé dans un état voisin puis il évolue vers un autre état qui représente la solution de manière contrôlée et assez lente. La solution trouvée n'est autre qu'une solution approchée à la solution exacte **Ezratty (n.d.)**. De nombreux chercheurs affirment que ce type d'ordinateurs est mi-quantique, mi-classique et qu'il n'a pas encore fait ces preuves en pratique **Gondran & Gondran (2020)**.

### 1.5.2 Les ordinateurs quantiques à portes logiques

Cette deuxième catégorie, utilise des portes quantiques pour manipuler les qubits et elle est capable d'exécuter n'importe quel algorithme quantique **Ezratty (n.d.)**. Elle apporte un gain de puissance et de vitesse par rapport aux simulateurs quantiques. Actuellement, l'utilisation de ces ordinateurs est limitée par le nombre de qubits disponible (environ une centaine de qubits) et par le bruit quantique qui influence sur la qualité des résultats obtenus **Gondran & Gondran (2020)**. On peut distinguer des ordinateurs quantiques sans code d'erreur qui sont généralement utilisés lorsque les problèmes étudiés ne nécessitent pas des solutions exactes. Cependant, lorsqu'on essaye de résoudre un problème tel que la décomposition en nombre premier, l'intégration d'un code de correction d'erreur est primordial pour pouvoir trouver des résultats corrects **Gondran & Gondran (2020)**.

### 1.5.3 Langage de programmation et Kits de développement logiciels

Un kit de développement représente un ensemble d'outils permettant le développement et la création des programmes quantiques ainsi que l'exécution de ces programmes sur des simulateurs pour pouvoir les tester. Ils offrent également la possibilité de tester ces programmes sur de vrais dispositifs quantiques disponibles sur le cloud **Zouambi (2019)**. Dans ce qui suit nous allons présenter les plus connus :

1. **Forrest** qui est une plateforme logicielle quantique développée par Rigetti, qui comprend le langage de programmation quantique PyQuil open-source intégré dans le langage classique Python, pour construire, analyser et exécuter des programmes quantiques. Elle comporte aussi des simulateurs locaux, des bibliothèques comportant des programmes quantiques et des solutions cloud.
2. **Quantum Information logiciel kit - Qiskit** qui comporte une collection d'outils Open Source permettant de réaliser des circuits quantiques avec différents langages de programmation Python, JAVA et SWIFT. Elle permet de tester les circuits sur des simulateurs locaux ou sur le cloud.
3. **Ocean** qui fournit une collection d'outils Open Source permettant d'exécuter des programmes écrits en Python sur les ordinateurs quantiques D-wave, des simulateurs ou des ordinateurs classiques.

Différents langages de programmation ont été développés pour écrire des algorithmes quantiques avec des instructions de haut niveau, puis de traduire ces instructions vers des instructions physiques qui s'exécutent sur un processeur quantique. La majorité de ces langages de programmation sont indépendants des architectures matérielles **Isaac Chuang (2002)**. Nous pouvons distinguer deux types de langages de programmation :

1. Les langages procéduraux ou impératifs basés sur l'utilisation des instructions afin de modifier l'état d'un système global ou d'une variable. Un bon nombre de ces langages existent et les plus connus sont : Q sharp, Q langage et QCL (Quantum Computation langage) **Zouambi (2019)**.
2. Les langages fonctionnels : QFC et QPL, Quantum lambda calcul et QML.

#### 1.5.4 Simulation des circuits quantique sur les ordinateurs classiques

Les simulateurs jouent un rôle essentiel dans le développement des ordinateurs quantiques et de leurs algorithmes. Leur mise en œuvre est confrontée à des défis en termes d'évolutivité et de faisabilité. Au niveau le plus bas, un simulateur peut être utilisé pour simuler le fonctionnement des portes matérielles quantiques natives afin de fournir les sorties attendues d'un ordinateur quantique qui peut à son tour être utilisé pour aider à vérifier le matériel. Au niveau le plus élevé, un simulateur peut suivre le calcul algorithmique logique et l'état des qubits logiques. Les simulateurs peuvent modéliser l'effet du bruit pour différentes technologies matérielles. Cela aide les concepteurs d'algorithmes à prévoir les effets du bruit sur les performances des algorithmes quantiques avant qu'il n'existe de machines capables de les exécuter. Ces capacités de simulation seront particulièrement importantes pour les systèmes NISQ dont l'absence de support signifie que les effets du bruit auront un impact fondamental sur les performances et le succès des algorithmes **National Academies of Sciences & Medicine (2019)**.

Le défi fondamental de la simulation est la rapidité de la mise à l'échelle de l'espace d'état. Puisqu'une opération de porte peut être implémentée sur un ordinateur classique par une séquence de multiplications matrice-vecteur. Cependant, la taille de la fonction d'onde à valeur complexe représentant l'état d'un ordinateur quantique avec  $N$  qubits croît par  $2^N$  bits. Cela signifie que le matériel QC avec un seul qubit supplémentaire possède un espace d'état double. Très rapidement, l'espace devient trop grand pour être simulé de manière satisfaisante, même sur le plus grand super-ordinateur classique. Actuellement, les super-ordinateurs sont capables de simuler des systèmes de l'ordre de 50 qubits. Le tableau 1.2 présente les simulateurs quantiques les plus connus ainsi que le nombre de qubits disponibles et le type d'exécution locale (sur la machine). **LaRose (2019)**.



simulateur		Nombre de qubits	Local
<b>Qiskit</b>	ibmq qasm simulator	32 qubits	NON
	simulator extended stabilizer	63 qubits	OUI
	local qasm simulator	32 qubits	OUI
<b>pyQuil</b>		30 qubits	NON, il ne peut pas être exécuté sur un ordinateur d'utilisateur local, mais seulement sur le cloud
<b>ProjectQ</b>		6	Oui
<b>QDK simulator</b>		30 qubits	Oui
<b>Qsim</b>		20/40 qubits	Oui
<b>Qubit Toaster</b>		Il dépend de performance de la machine utilisée	Oui

TABLE 1.2 – Les simulateurs des circuits quantiques

## 1.6 Enjeux et défis liés à la réalisation d'un système quantique

Il existe aujourd'hui des ordinateurs classiques suffisamment performant avec un taux de fiabilité de 100%, chose que nous n'avons pas encore atteint avec les ordinateurs quantiques. Certes ces ordinateurs existent, mais leurs utilisations est restreinte par plusieurs contraintes comme :

1. **La décohérence** qui représente le phénomène de perte de l'information quantique. En effet, les qubits sont très sensibles aux interactions avec l'environnement et si le système quantique n'est pas parfaitement isolé, le qubit tend rapidement à perdre ses propriétés quantiques **Umeano (2021)**. Pour remédier à ce problème, les qubits doivent être dans un environnement bien isolé et être manipulés avec un très grand degré de contrôle. Chose qui n'est pas facile, car lorsqu'on essaye de faire du calcul, nous serons obligés de manipuler les qubits et une telle interaction peut entraîner

des perturbations d'énergie. La décohérence représente une véritable préoccupation lors de la conception des algorithmes quantiques **Zouambi (2019)**.

2. **Le nombre de qubits disponibles sur les ordinateurs quantiques.** Actuellement, on dispose d'ordinateurs quantiques avec un nombre de qubits entre 50 et 100 qubits **EL BAHLOUL (2008)**. Cette contrainte rend la conception et l'exécution de certains algorithmes difficile voir impossible pour des instances de grande taille.
3. **Le taux d'erreurs des portes quantiques.** Selon Anthony Leverrier *Une porte fait ce qu'il faut 99,9% du temps, et 0,1% du temps, elle commet une petite erreur. Au bout d'à peu près 1000 portes, il y aura une erreur quelque part dans le circuit*. Ce qui induit la dégradation de la qualité des résultats des calculs et entraînent des erreurs de mesure. Il est à noter que les taux d'erreurs pour un opérateur à un seul qubit sont entre  $10^{-3}$  et  $10^{-6}$ , et pour un opérateur à 2 qubits sont entre  $10^{-2}$  et  $10^{-3}$  **National Academies of Sciences & Medicine (2019)**.
4. **L'incapacité de copier ou de stocker l'information selon le théorème de non-clonage quantique.** Il est impossible de copier des états quantiques et par conséquent, nous ne pouvons pas garder ou stocker des copies des résultats intermédiaires obtenus lors du calcul. Ceci rend la tâche d'élaborer un algorithme quantique plus difficile par rapport à un algorithme classique **National Academies of Sciences & Medicine (2019)**. Afin de pallier ces limites, plusieurs techniques ont été proposées tel que les codes de correction d'erreurs qui représentent un élément essentiel pour pouvoir protéger l'information quantique. Ainsi que les RAMs quantique pour pouvoir stocker l'information quantique, mais pour le moment cette idée reste un sujet de recherche.

Un ordinateur quantique universel tolérant aux pannes qui peuvent résoudre efficacement des problèmes tels que la factorisation d'entiers et la recherche de bases de données non structurées nécessite des millions de qubits avec de faibles taux d'erreur et de longs temps de cohérence et les progrès expérimentaux vers un tels dispositifs nécessitera des années voir des décennies de recherche. Actuellement, nous disposons d'ordinateurs quantiques bruyants à échelle (NISQ) qui comportent des centaines de qubits bruyants, c'est-à-dire des qubits qui ne subiront pas de correction d'erreurs et qui, par conséquent, effectue des opérations imparfaites dans un temps de cohérence limité **Bharti et al. (2021)**. Actuellement, plusieurs travaux ont été proposés dans le cadre de la recherche d'un avantage quantique avec ces dispositifs Des algorithmes ont été proposés pour des applications dans diverses disciplines allant de la physique à l'apprentissage automatique, la chimie quantique et l'optimisation combinatoire afin d'exploiter les ressources limitées disponibles pour effectuer des tâches classiques difficiles **Bharti et al. (2021)**.

## 1.7 Conclusion

Dans ce chapitre, nous avons présenté les concepts nécessaires pour la compréhension de la théorie de l'information quantique. Nous avons aussi introduit les différents modèles de calcul quantique ainsi que les algorithmes quantiques. Nous avons présenté également tous les aspects liés à l'informatique quantique : les dispositifs quantiques disponibles, les simulateurs quantiques, les langages de programmations et les kits de développements. En outre, nous avons abordé les défis et les enjeux liés à la réalisation d'un système quantique.

À l'heure actuelle, l'informatique quantique n'est pas encore au point d'être exploitée en pratique. Les ordinateurs quantiques disponibles souffrent des problèmes de décohérence et des taux d'erreurs. Mais les performances théoriques des algorithmes quantiques ainsi que les avantages qu'ils offrent tel que la possibilité d'explorer un grand espace de recherche avec un parallélisme massive nous encourage à poursuivre la recherche dans ce domaine et à étudier l'impact de l'exploitation du calcul quantique pour la résolution des problèmes d'optimisation combinatoire surtout que jusqu'à nos jours, nous ne disposons pas encore des algorithmes efficaces pour la résolution de ces derniers.

# Chapitre 2

## Optimisation combinatoire quantique

### 2.1 Introduction

L'optimisation combinatoire représente un axe de recherche très actif combinant plusieurs domaines tel que les mathématiques discrètes, la recherche opérationnelle et l'informatique. L'importance de ce domaine se justifie d'une part par la difficulté de la résolution des problèmes d'optimisation combinatoire (POCs) et d'autre part par la large gamme de problèmes qui peuvent être modélisés comme des POCs. Nous pouvons citer par exemple les problèmes de l'industrie, les problèmes de la finance et de la vie quotidienne. La majorité de ces problèmes sont considérés comme des problèmes NP-difficile. Principalement, la résolution de ces problèmes consiste en la recherche de l'optimum d'une fonction parmi un ensemble fini et dénombrable de solutions, mais cet ensemble risque d'être trop grand, ce qui rend la résolution des POCs plus difficile malgré que la formulation de ces problèmes soit facile.

Généralement, les méthodes exactes qui consistent à énumérer toutes les solutions possibles pour la résolution de ce type de problèmes sont coûteuses en termes de temps et d'espace mémoire. Cette limite a permis la naissance d'un nouveau type de méthodes appelées les méthodes approchées qui cherchent à trouver des solutions efficaces qui assure un bon compromis entre la qualité de la solution et le temps d'exécution.

Dans ce chapitre, nous allons présenter les nouvelles approches pour la résolution des POCs basées sur le calcul quantique. Dans un premier temps, nous allons commencer par introduire les différents concepts liés à l'optimisation combinatoire ainsi qu'à la théorie de la complexité. Par la suite, nous allons proposer une classification de toutes les méthodes de résolution de POCs quantique ainsi qu'une présentation des méthodes les plus connus.

## 2.2 Les problèmes d'optimisation combinatoire - Définition et Complexité

Un problème d'optimisation combinatoire (POC) consiste en la recherche d'une ou de plusieurs solutions dans un espace dénombrable et fini de solutions, appelé espace de recherche ou de décisions, afin de maximiser ou minimiser une fonction qui mesure le coût de la solution, appelé fonction objectif, en respectant certaines contraintes **Zouambi (2019)**.

Plus formellement, un problème d'optimisation combinatoire est défini comme suit. Étant donné un espace de solutions possibles  $S$ , et une fonction objectif  $f : S \rightarrow R$ , il s'agit de trouver la solution  $s^* \in S$  tel que :

- $f(s^*) \leq f(s_i) \forall s_i \in S$  pour un problème de minimisation.
- $f(s^*) \geq f(s_i) \forall s_i \in S$  pour un problème de maximisation.

.

Les problèmes d'optimisation combinatoire peuvent être classés en deux classes : les problèmes mono-objectif avec une seule fonction objectif et les problèmes multi-objectifs qui regroupe  $k$  fonctions objectives. Malgré que la formulation de ces problèmes est souvent facile, leur résolution reste très difficile. En effet, la majorité de ces problèmes appartient à la classe NP-difficile et on ne possède pas encore d'algorithmes efficaces pour les résoudre **Zouambi (2019)**.

La théorie de la complexité consiste à étudier et à classer des problèmes exprimés d'une façon mathématique en fonction de la difficulté des algorithmes utilisés pour les résoudre **Layeb (2010)**. Nous pouvons distinguer deux types de complexité : la complexité spatiale et la complexité temporelle. La complexité spatiale représente l'espace requis pour l'exécution de l'algorithme en fonction de la taille de l'instance utilisée. Alors que la complexité temporelle représente le temps mis par un algorithme pour s'exécuter en fonction de la taille des données d'entrée. Elle est exprimée par la fonction  $T(n)$  où  $n$  représente la taille du problème **Zouambi (2019)**.

Dans ce qui suit, nous allons présenter les classes de complexité des problèmes de décision qui se définit comme un problème dont la réponse est soit vrai, soit faux. Il est à noter qu'il existe des problèmes qui ne sont pas des problèmes de décision, mais des problèmes d'optimisation. Afin de pouvoir étudier ces problèmes dans le cadre de la théorie de complexité, il suffit de les transformer en problèmes de décision en les formulant de la manière suivante : étant donné une fonction objective  $f$ , un espace de solutions  $S$ , un seuil  $\alpha$ . Dans le cas de la minimisation (resp maximisation), est-il possible de trouver une solution  $x \in S$  telle que  $f(x) \leq \alpha$  (resp  $f(x) \geq \alpha$ ) ?

Selon la classification de Garey et Johnson (1979) nous pouvons distinguer les classes de complexité suivantes :

1. **La classe des problèmes polynomiaux  $P$**  . Cette classe regroupe les problèmes de décision qui peuvent être résolus sur une machine déterministe avec un temps polynomial. Ces problèmes peuvent être résolus efficacement.
2. **La classe des problèmes non déterministe  $NP$**  . Cette classe englobe des problèmes dont la solution peut être vérifiée facilement par un algorithme non déterministe en un temps polynomial par rapport à la taille de l'instance, mais leur résolution reste difficile et nécessite la vérification de tous les cas possibles. Dans la majorité des cas le temps d'exécution de ces algorithmes est exponentiel.
3. **La classe  $NP$ -complet**. cette classe représente un sous ensemble de la classe  $NP$ , contenant les problèmes  $NP$  les plus difficiles et tous les problèmes de la classe  $NP$  se ramenant à celui-ci via une réduction polynomiale. Théoriquement, il est prouvé que si nous arrivons à trouver un algorithme de complexité polynomiale pour résoudre d'un problème  $NP$ -Complet, toutes les solutions aux problèmes de cette classe serait facile à trouver (Théorème de Cook).
4. **La classe  $NP$ -difficile**. Cette classe regroupe les problèmes d'optimisation combinatoire qui appartiennent à la classe  $NP$ -complet.

En plus des classes de complexité citées auparavant, il existe des classes de complexité permettant l'étude de la difficulté des algorithmes quantiques **Montanaro (2016)**. Dans cette catégorie nous pouvons citer :

1. **La classe  $BQP$  (Bounded error quantum polynomial time)** qui décrit la classe des problèmes qui peuvent être résolu en un temps polynomial sur un ordinateur quantique **Montanaro (2016)**. Cette classe a été introduite en 1993 avec l'apparition des premiers algorithmes quantiques. Elle est connue pour la résolution des problèmes qui ne sont pas encore résolus d'une manière efficace sur un ordinateur classique comme l'algorithme de Shor qui permet de factoriser un nombre entier avec une complexité égal à  $\log(n)^3$ . Actuellement, sur un ordinateur classique, il n'existe pas un algorithme de complexité polynomiale qui peut résoudre ce problème. Aussi l'algorithme de Grover qui permet de trouver un élément  $n$  dans une liste avec une complexité  $\sqrt{N}$ , tel que  $N$  représente le nombre d'éléments de la liste, alors que sur un ordinateur classique, on doit parcourir  $N$  éléments. Il est démontré que la classe  $P \in BQP$  mais la question  $P = BQP$  reste une question ouverte **Vert (2021)**. La figure 2.1 représente les questions ouvertes sur les liens entre la classe de complexité  $BQP$  et la théorie de complexité classique.

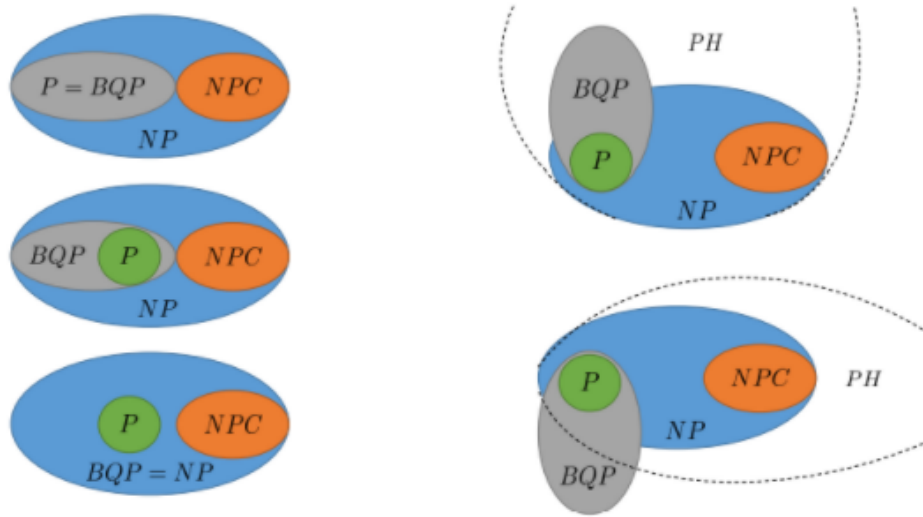


FIGURE 2.1 – Les liens possibles entre le calcul quantique et la théorie de la complexité Vert (2021)

2. **La classe QMA** qui définit l'ensemble des problèmes qui sont vérifiables en un temps polynomial sur un ordinateur quantique Umeano (2021).

## 2.3 Résolution des POCs sur un ordinateur quantique

Certains algorithmes quantiques comme les algorithmes variationnels et le recuit quantique nécessitent la reformulation des POCs sous forme d'un Hamiltonien pour pouvoir les résoudre. Un Hamiltonien est défini comme une matrice permettant de décrire l'énergie d'un système quantique. Tous d'abord nous devons modéliser notre problème sous la forme QUBO puis le mapper vers un Hamiltonien du modèle Ising 2.2. Dans ce qui suit nous allons présenter les concepts théoriques essentiels pour la formulation des POCs. Commençons par le modèle physique Ising et la forme QUBO, ensuite nous allons présenter comment la fonction objectif d'un POC est reformulée.

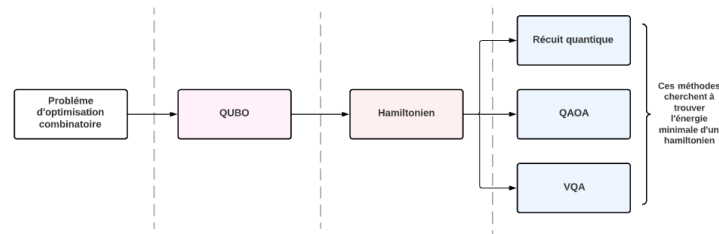


FIGURE 2.2 – Reformulation d'un POC

L'optimisation binaire quadratique sans contrainte (en anglais Quadratic unconstrained Binary Optimisation QUBO) est un modèle unifié pour représenter un large gamme

de POCs, tel que le problème du voyageur de commerce, le problème du bin packing, les problèmes de flux de réseaux, de coupe maximale, de clique maximale, de vertex cover et d'autres problèmes de graphes **Lewis & Glover (2017)**. Formellement on définit un problème QUBO par une matrice  $Q$  carré  $N \times N$  symétrique à coefficient réel  $q_{ij}$  et un vecteur  $x$  qui comporte des variables binaires **Vert (2021)** tel que la fonction à minimiser/maximiser est :

$$f(Q, x) = \sum_j Q_{jj}x_j + \sum_{i < j} Q_{ij}x_i x_j \quad (1)$$

tel que  $x_i \in \{0, 1\}$  et  $i \in N$ . Le problème QUBO cherche à trouver le vecteur  $x$  à valeur binaire minimisant la fonction  $f(x)$ , Nous pouvons l'exprimer aussi de la manière suivante :

$$\min_{\forall x \in \{0,1\}^n} x^T Q x$$

Notons que pour les problèmes d'optimisation avec des contraintes, la fonction de coût est complétée par des termes de pénalité qui pénalisent les solutions qui ne correspondent pas à des solutions valides ou faisables au lieu d'imposer explicitement des contraintes au sens classique. les pénalités sont formulées de sorte qu'elles soient égales à zéro pour les solutions réalisables et égales à une certaine quantité positive de pénalité pour les solutions infaisables **Glover et al. (2022)**. Le tableau 2.1 représente les contraintes les plus utilisés et la fonction de pénalité équivalente. Le choix des valeurs de pénalité est très important. En effet, choisir une valeur de pénalité trop importante peut entraver le processus de résolution, car les termes de la pénalité inhibent l'évaluation de la fonction objectif original, ce qui rend difficile de distinguer la qualité d'une solution d'une autre. D'autre part, une valeur de pénalité trop basse peut compromettre la recherche de solutions réalisables **Glover et al. (2022)**.

Contrainte classique	pénalité équivalente
$x = y$	$P(x + y - 2xy)$
$x + y = 1$	$P(1 - x - y - 2xy)$
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y - xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$

TABLE 2.1 – Les contraintes classiques les plus connus et leurs pénalités équivalentes.

Les ordinateurs quantiques dont nous disposons aujourd'hui nécessitent une certaine compréhension des concepts de la physique quantique. C'est pourquoi nous allons décrire dans cette partie le modèle d'Ising utilisé pour résoudre les problèmes QUBO. Le



modèle d'Ising, proposé pour la première fois par Ernst Ising et Wilhelm Lenz, explique l'interaction des molécules d'un matériau magnétique dans un champ magnétique externe **Zahedinejad & Zaribafiyan (2017)**. Sous une forme mathématique abstraite, un modèle d'Ising est constitué d'un ensemble de spins  $V$ , chacun prenant une valeur de  $s_i \in \{-1, 1\}$ . En considérant  $L$  comme un ensemble d'interactions par paire entre les spins. Nous pouvons représenter l'énergie  $E(s)$  du système de spins pour une configuration  $s \in \{-1, 1\}^{|V|}$  par l'Hamiltonien suivant **Zahedinejad & Zaribafiyan (2017)** :

$$E(s) = \sum_{(i,j) \in L} J_{i,j} s_i s_j + \sum_{i \in V} h_i s_i \quad (2)$$

- $J$  représente les matrices d'interaction de couplage de spin  $s_i$  et  $s_j$ .
- $h_i$  représente le champ magnétique externe sur le spin  $s_i$ .

Les algorithmes quantiques comme le recuit quantique par exemple cherche à trouver l'état fondamental qui correspond à l'énergie minimale du Hamiltonien en mappant un problème d'optimisation vers un Hamiltonien du modèle Ising. Nous pouvons le résoudre en exploitant les algorithmes quantiques qui ont comme objectif la recherche d'un état fondamental, c.-à-d. l'état **correspondant à l'énergie minimale du système Zahedinejad & Zaribafiyan (2017)**.

En utilisant les deux modèles QUBO et Hamiltonien de Ising exprimé par les équations (1) et (2) on constate qu'il est possible de trouver facilement la forme QUBO du modèle Ising comme suit :

$$f(x_1, x_2, x_3, \dots, x_n) = c_0 + \sum_{j=1}^n c_j x_j + \sum_{1 \leq j \leq i \leq n} d_{ij} x_i x_j \quad (3)$$

Avec un petit changement de variable  $s_i = 2x_i - 1$ , l'équation précédente serait équivalente à l'équation (2). En considérant cette correspondance, la résolution d'un problème QUBO devient équivalente à la recherche de l'état fondamental d'un modèle d'Ising. Par conséquent, nous pouvons utiliser l'ordinateur quantique pour trouver l'état qui correspond à l'énergie la plus basse du Hamiltonien quantique et donc résoudre le problème QUBO **Zahedinejad & Zaribafiyan (2017)**.

Nombreux problèmes d'optimisation ont été exprimés sous la forme Ising **Lucas (2014)** tel que le problème du voyageur de commerce (TSP), les problèmes de coloration de graphe, de partitionnement et de planification. En utilisant cette formulation, nous pouvons explorer des nouvelles approches pour la résolution des POCs. Dans ce qui suit, nous allons présenter un exemple de reformulation de problème de coloration de graphe sous la forme QUBO. Le problème de coloration de graphe est un problème NP-difficile qui peut être défini comme un graphe non orienté  $G = (V, E)$  et un ensemble de  $n$  couleurs. Sa-

chant qu'on ne dispose que de  $K$  couleurs possibles, le problème consiste à colorer chaque sommet du graphe avec une couleur spécifique, de sorte qu'aucune arête ne relie deux sommets de la même couleur.

Ce problème peut être modélisé sous la forme QUBO comme suit :  
Considérons la variable  $x_{ij}$  tel que :

$$x_{ij} = \begin{cases} 1, & \text{si la couleur } j \text{ est attribuée au noeud } i \\ 0 & \text{sinon} \end{cases}$$

Puisque chaque nœud doit être coloré, nous avons les contraintes suivantes :

$$\sum_{j=1}^K x_{ij} = 1 \quad i = 1, \dots, n \quad (2.1)$$

où  $n$  est le nombre de nœuds du graphe. Il reste à vérifier si une coloration donnée est réalisable, autrement dit vérifier si les nœuds adjacents sont affectés à des couleurs différentes. Ceci est assuré en imposant les contraintes suivantes :

$$x_{ip} + x_{jp} \leq 1 \quad p = 1, 2, \dots, K \quad \forall (i, j) \in E. \quad (2.2)$$

Afin d'obtenir la formulation QUBO du problème, on doit mapper la contrainte (2.1) en des pénalités quadratiques à ajouter à la fonction objectif et la deuxième contrainte peut être reformulée de la même façon qu'on a reformulé  $x + y \leq 1$  dans le tableau 2.1. Comme résultat, nous allons avoir :

$$F = P \sum_{i=0}^n (1 - \sum_{c=1}^K x_{ic})^2 + P \sum_{(i,j) \in E} \sum_{c=1}^K x_{ic} x_{jc} \quad (2.3)$$

Afin d'obtenir le Hamiltonien du modèle Ising, il suffit de remplacer les  $x_{ic}$  par  $\frac{s_i+1}{2}$  tel que  $s_i \in \{-1, 1\}$ .

## 2.4 Classification des méthodes quantique pour la résolution des POCs

La branche de l'optimisation combinatoire quantique représente un domaine de recherche très actif qui comporte une large gamme d'algorithmes, d'approches et de principes. Ces méthodes peuvent être classées selon la nature de la solution trouvée, approchée ou exacte. Aussi ils peuvent être classés selon la façon dont on formule notre problème **Zahedinejad & Zaribafiyani (2017)**. La classification, que nous proposons dans ce mé-

moire, est basée sur la nature des dispositifs utilisés pour la résolution des POCs. En effet, en commençant notre recherche bibliographique, nous avons constaté qu’une des différences principales réside dans les méthodes trouvées est le type d’ordinateur utilisé pour les exécuter. Le schéma suivant résume les méthodes quantiques de résolution des POCs 2.3 :

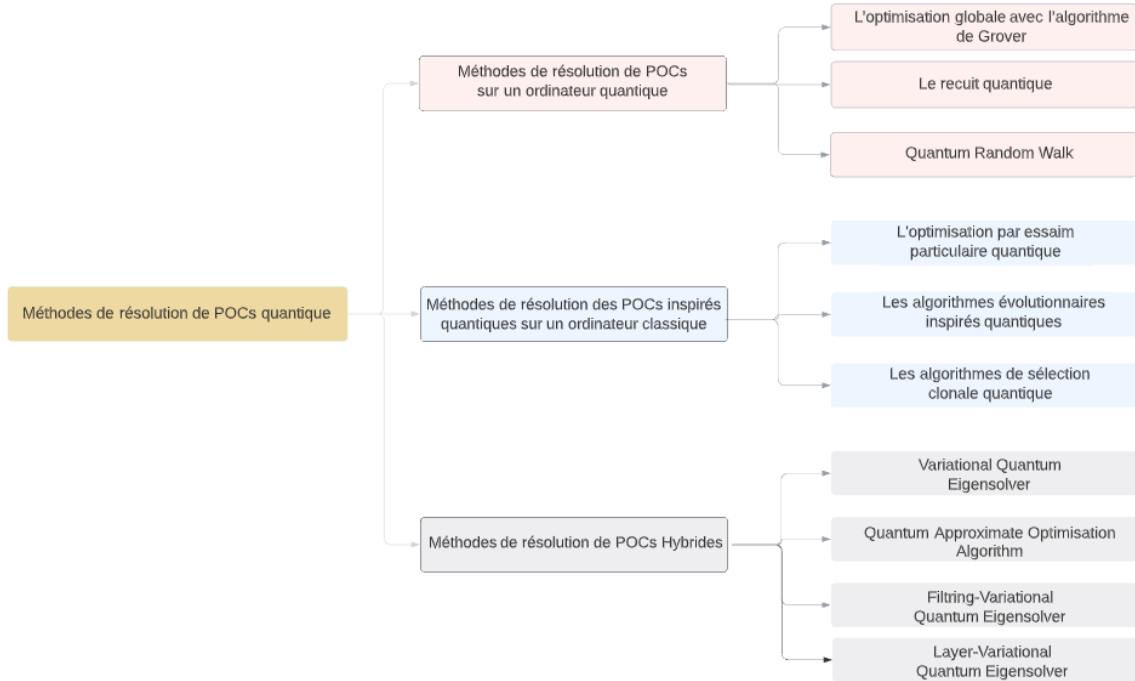


FIGURE 2.3 – Classification des méthodes de résolution de POCs

### 2.4.1 Méthodes de résolution des POCs sur un ordinateur quantique

Cette catégorie regroupe les méthodes quantiques qui s’exécutent sur un vrai ordinateur quantique. Soit sur un ordinateur à portes logiques comme les méthodes d’optimisation globale basée sur l’algorithme de recherche de Grover, soit sur un ordinateur à recuit quantique comme l’ordinateur de D-Wave. La majorité des algorithmes quantiques d’optimisation globale utilisent l’algorithme de recherche de Grover comme une procédure de recherche **Gondran & Gondran (2020)**. Cet algorithme permet d’accélérer la résolution de n’importe quel problème qui peut être exprimé comme un problème de recherche avec une vitesse quadratiquement meilleure par rapport au meilleur algorithme classique **Grover (1996); Boyer et al. (1998)**.

L’algorithme de recherche de Grover s’est avéré optimal pour la recherche dans une base de données non structurée **Grover (1996)**. Il a été largement utilisé dans le contexte des algorithmes d’optimisation quantique générale. **Durr & Hoyer (1996)** ont été les

premiers à présenter un algorithme quantique permettant de trouver le minimum d'une base de données non structurée. Inspirés par ce travail, d'autres chercheurs ont également proposé des algorithmes quantiques pour les problèmes d'optimisation. Dans ce qui suit, nous allons passer en revue les étapes fondamentales de cet algorithme ainsi que son adaptation pour la résolution des problèmes d'optimisation combinatoire. L'algorithme a été introduit en 1996 par L.Grover dans **Grover (1996)** pour la recherche d'un élément  $x$  dans une liste avec  $N$  éléments. Il se compose principalement de trois parties :

1. **Initialisation** . Cette étape consiste à créer une superposition uniforme où tous les états possibles possèdent la même probabilité d'être choisis. Soit un espace de recherche,  $S = \{0,1\}^n$  l'état de superposition uniforme de cet espace est  $\frac{1}{\sqrt{N}} \sum_{x=0}^{2^n-1} |x\rangle$  où la probabilité d'être dans n'importe quel état est  $\frac{1}{\sqrt{N}}$  et  $N = 2^n$  [3]. Afin de réaliser cette superposition uniforme, l'algorithme utilise l'opérateur de Hadamard décrit dans la section précédente **Grover (1996)**.
2. **L'oracle V** . C'est la partie du circuit qui change selon le problème traité. Elle consiste à distinguer et à marquer l'état ou les états correspondants à la solution avec une valeur négative en faisant tourner la phase de  $\pi$  radians et garder le reste des états inchangés. Il est possible d'exprimer cette opération comme suit :  $|x\rangle = (-1)^{o(x)}|x\rangle$  où  $o(x)$  est égal à 1 pour les états solutions et 0 sinon **Grover (1996)**.
3. **L'opérateur de diffusion D**. Après que l'oracle marque les solutions avec un signe négatif, l'opérateur de diffusion amplifie les amplitudes de ces états, en utilisant le principe de l'inversion autour de la moyenne. Ce principe augmente l'amplitude des valeurs qui sont inférieures à la moyenne, simultanément, il réduit la valeur de l'amplitude supérieure à la moyenne **Grover (1996)** comme exprimé dans la figure 2.4.

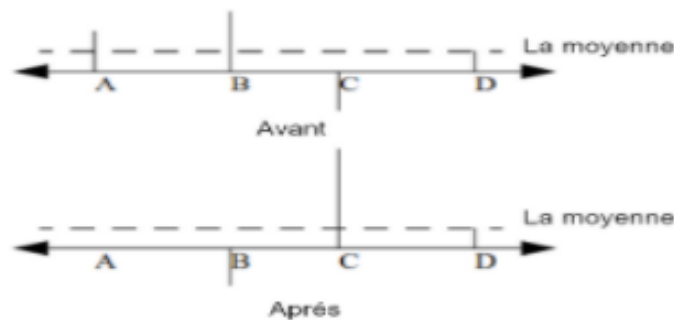


FIGURE 2.4 – L'inversion autour de la moyenne tirée de **Durr & Hoyer (1996)**

L'application de l'oracle V et l'opérateur de diffusion D permet d'augmenter l'amplitude de l'état visé comme décrit dans la figure 2.5.

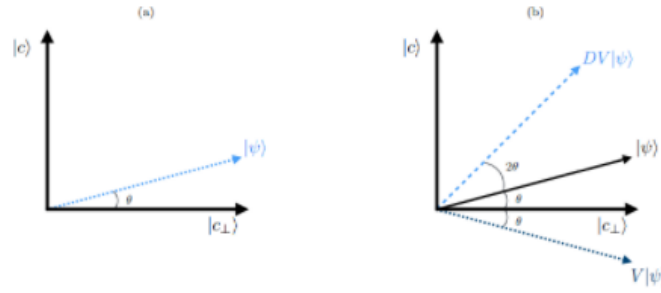


FIGURE 2.5 – Représentation géométrique de l'algorithme de recherche de Grover dans un espace à deux dimensions. (a) Représentation de l'état initial dans l'espace bidimensionnel des états cible ( $|c_i\rangle$ ) et non cible ( $|c_\perp\rangle$ ). (b) Action des opérateurs  $D$  et  $V$  sur l'état initial **Zahedinejad & Zaribafiyani (2017)**.

Le processus commence par une solution initiale (obtenue par une borne supérieure (problème de minimisation), soit une solution réalisable quelconque). Ensuite, la tâche de l'algorithme de Grover consiste à trouver toute autre solution dont le coût est meilleur que la solution initiale. Si le résultat de la recherche est positif, la solution initiale est remplacée par la nouvelle solution trouvée et l'itération est répétée jusqu'à ce que certains critères soient remplis. Dans le cas où le processus de recherche échoue, soit nous abandonnons l'algorithme, soit nous affinons le bloc de recherche de Grover en utilisant un sous-programme de planification de la rotation. **Zahedinejad & Zaribafiyani (2017)**.

Certes, l'algorithme de Grover ne change pas la nature NP-difficiles des problèmes étudiés, mais il permet d'avoir une amélioration importante, surtout pour les grandes instances. Cet algorithme a été utilisé pour la résolution de nombreux problèmes d'optimisation combinatoire. **Vert (2021)** ont proposé un algorithme pour la résolution du problème de satisfiabilité des contraintes avec une amélioration quadratique par rapport aux algorithmes classiques. l'algorithme a été aussi adapté pour la résolution de  $R||C_{max}$  où ils ont modifié la partie d'initialisation classique de Grover qui génère un espace de recherche comportant tous les états possibles dans l'espace Hadamard par un circuit quantique permettant de générer toutes les solutions possibles avec leurs coûts **Lu & Marinescu (2007)**. Par ailleurs, **Bulger et al. (2003)** ont proposé une amélioration de l'algorithme de Grover appelé Grover Adaptive Search. Cet algorithme génère le nombre d'itérations nécessaire pour exécuter Grover quand on ne peut pas savoir le nombre de solutions possibles. Une méthode générale pour la résolution de n'importe quel problème d'optimisation écrit sous la forme QUBO ou la forme CPBO (optimisation combinatoire binaire avec contrainte) est proposé dans **Gilliam et al. (2021)**. **Mukherjee (2022)** ont proposé de changer la base algébrique pour pouvoir générer un espace de recherche comportant seulement les états possibles pour la résolution du problème de coloration de liste. Une méthode pour adapter l'oracle de Grover pour la résolution de problème de max clique **Haverly & López (2021)**. **Scherer et al. (2021)** ont proposé une méthode

pour encoder le problème de planification avec moins de qubits.

La majorité des travaux présentés ci-dessus utilisent l'algorithme de Grover qui apporte une amélioration par rapport aux algorithmes classiques. cependant, du fait du nombre de qubits disponibles, l'algorithme était testé seulement sur de petites instances qui sont considérées comme faciles pour les algorithmes classiques. Aussi, cet algorithme n'est pas adapté pour être exécuté sur les ordinateurs quantiques disponibles actuellement, car il représente une méthode exacte qui attend que les portes quantiques fassent le calcul sans erreurs, alors qu'avec le matériel quantique disponible ne peut effectuer un calcul sans erreurs.

## 2.4.2 Méthodes hybrides de résolution des POCs

Les ordinateurs quantiques disponibles aujourd'hui ne sont pas encore au point. En effet, pour pouvoir exécuter un algorithme quantique tel que l'algorithme de Grover ou Shor nous aurions besoin d'un grand nombre de qubits, de portes logique et des algorithmes de correction d'erreur et ceci nécessite des années voir des décennies. Afin de pouvoir tirer profit et explorer les processus quantiques actuels, en tenant compte des limites de ces derniers, un autre type d'algorithmes ont été proposés : les algorithmes variationnels hybrides quantiques. Il existe plusieurs types d'algorithmes variationnels, dans ce qui suit nous allons présenter, les algorithmes utilisés pour la résolution des POCs.

### 2.4.2.1 Variational Quantum Eigensolver - VQE

**Peruzzo et al. (2014)** ont introduit un algorithme variationnel hybride permettant d'estimer les énergies des états fondamentaux des molécules décrites par un Hamiltonien dans le contexte de la chimie quantique et comme les problèmes d'optimisation combinatoire de la forme QUBO peuvent être exprimés sous forme d'un Hamiltonien, cet algorithme a également été utilisé pour les résoudre **Moll et al. (2018)**. Il est considéré comme l'un des meilleurs candidats pour le calcul quantique à échelle intermédiaire bruyant (NISQ), en raison du faible nombre de qubits et de portes généralement nécessaires pour l'implémenter **Leonidas (2021)**. Cet algorithme permet de trouver une approximation de la valeur propre minimale  $\lambda_{min}$  ainsi que son vecteur propre associé  $|\psi(\theta)_{min}\rangle$  **Leonidas (2021)**, tel que :

$$\lambda_{min} = \langle \psi(\theta)_{min} | Hq | \psi(\theta)_{min} \rangle.$$

Il comporte quatre étapes principales (4.2). D'abord, sur le processeur quantique, un état quantique  $|\psi(\theta)_i\rangle$  est généré grâce à l'application d'une séquence de portes. Cet état est paramétré par un ensemble d'angles  $\theta = (\theta_1, \theta_2, \dots, \theta_n)$ . La séquence des portes est représentée par l'opérateur  $U(\theta)$  qui permet d'explorer l'espace des états quantiques **Leonidas (2021)**. La deuxième étape consiste en la détermination de la valeur du Hamiltonien  $Hq$

décomposé en un ensemble de matrice Pauli **Moll et al. (2018)**. Ensuite, le calcul de l'énergie du Hamiltonien exprimée comme suit :  $E_q(\theta) = \langle \psi(\theta)_q | H_q | \psi(\theta)_q \rangle$ . Finalement, l'algorithme d'optimisation classique est utilisé pour générer un nouveau paramètre  $\theta$ . La figure 4.2 représente un schéma général qui décrit le processus de l'algorithme VQE.

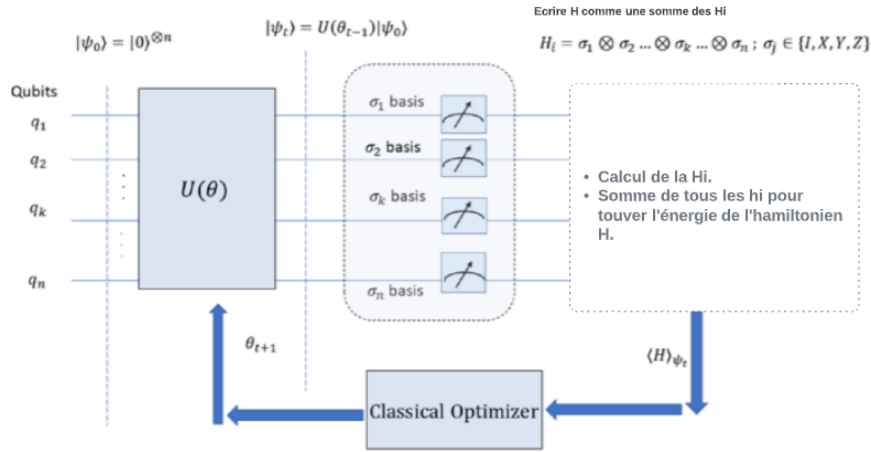


FIGURE 2.6 – Schéma général présentant le processus d'exécution des algorithmes quantique hybride VQE **Qiskit & Pattanayak (n.d.)**

Cette méthode compte parmi les meilleures candidates pour être exécutée sur les dispositifs NISQ. La méthode a été proposée pour la résolution des problèmes de la chimie et la résolution des nombreux POCs tel que le problème du Job SHOP **Amaro et al. (2022a)**, du TSP et du problème d'affectation quadratique QAP **Khumalo et al. (2021)** et le problème de coupe maximum **Liu et al. (2022)**. l'utilisation de cette méthode donne de bons résultats pour certains problèmes, mais des solutions non faisables aussi pour d'autres. Par ailleurs, la méthode a été utilisée seulement pour la résolution de petites instances à cause de la limites lié au matériel. **Liu et al. (2022)** ont proposé une amélioration de la méthode et ont utilisé plusieurs couches de VQE au lieu d'une seule. En 2021, un autre algorithme appelé Filtring-VQE a été proposé qui utilise le même circuit de VQE combiné avec des opérateurs de filtrage pour parvenir à une convergence plus rapide et plus fiable vers la solution optimale et les cônes de causalité pour réduire le nombre de qubits requis sur un ordinateur quantique **Amaro et al. (2022b)**.

#### 2.4.2.2 Quantum Approximate Optimisation Algorithm - QAOA

L'algorithme d'optimisation approché quantique (en anglais Quantum Approximate Optimisation Algorithm QAOA ) est une métaheuristique utilisé pour la résolution des POCs sur un ordinateur à portes logiques **Rieffel et al. (2020)**. Cet algorithme a été proposé en 2014 par **Farhi et al. (2014)**. Tout comme l'algorithme VQE, cet algorithme est un algorithme variationnel hybride comportant deux parties : une partie classique

pour trouver les paramètres optimaux du circuit et une partie quantique permettant l'exploration de l'espace de Hilbert. Au lieu de générer un état quantique  $|\psi(\theta)\rangle$  par un seul opérateur unitaire  $U(\theta)$  comme VQE, l'algorithme QAOA est basé sur deux opérateurs unitaires  $U(C, \gamma)$  et  $U(B, \beta)$  utilisés pour la préparation d'un état quantique  $|\psi(\gamma, \beta)\rangle$  paramétré par les angles  $\gamma$  et  $\beta$  et d'un paramètre  $p \geq 1$  comme le montre la figure 2.7. L'entier  $p$  désigne la profondeur du circuit tandis que les opérateurs unitaires correspondent à certaines rotations appliquées sur les qubits **Leonidas (2021)**. l'opérateur  $U(C, \gamma)$  dépend du Hamiltonien  $C$  appelé Hamiltonien du coût décrivant la fonction de coût du problème traité et du paramètre  $\gamma$ . Le deuxième opérateur dépend du Hamiltonien  $B$  appelé le mixer Hamiltonien.

L'algorithme consiste en l'application d'une séquence alternée des opérateurs  $U(C, \gamma)$  et  $U(B, \beta)$   $p$  fois sur un état initial  $|+\rangle$ . Chaque itération  $p$  utilise différents paramètres  $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_p)$  et  $\beta = (\beta_1, \beta_2, \dots, \beta_p)$  de telle sorte que nous obtenons à la fin l'état quantique suivant :

$$|\psi_p(\gamma, \beta)\rangle = U(C, \gamma_1)U(B, \beta_1)U(C, \gamma_2)U(B, \beta_2)\dots U(C, \gamma_p)U(B, \beta_p)|+\rangle^{\otimes n}$$

Les paramètres  $\gamma$  et  $\beta$  sont optimisés à l'aide d'un optimiseur classique. l'objectif de l'optimiseur classique est de trouver les paramètres variationnels optimaux qui minimisent la valeur du Hamiltonien de coût **Farhi et al. (2014)**.

$$(\gamma^*, \beta^*) = \operatorname{argmin}_{\gamma, \beta} \langle \psi(\gamma, \beta) | H_C | \psi(\gamma, \beta) \rangle$$

Dans **Farhi et al. (2014)**, les auteurs ont montré que plus le nombre d'itérations  $p$  est grand, meilleure est la solution.

$$\lim_{p \rightarrow \infty} \operatorname{Max}_{\gamma, \beta} \langle \psi(\gamma, \beta) | H_C | \psi(\gamma, \beta) \rangle = \operatorname{Max} C(x)$$

$C(x)$  représente la fonction de coût

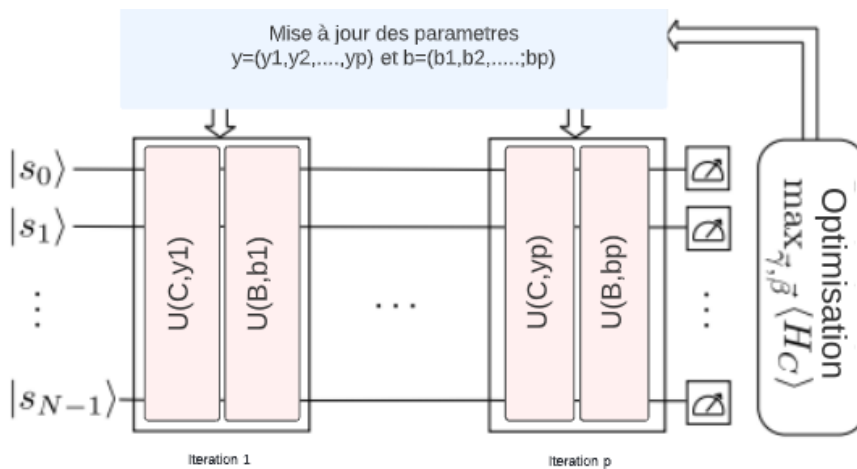


FIGURE 2.7 – Le processus d'exécution de l'algorithme de QAOA. **Rieffel et al. (2020)**



Le QAOA a été utilisé pour la résolution d’une large gamme des problèmes d’optimisation combinatoire, tels que le problème de couverture maximale de sommets **Cook et al. (2020)**, la résolution du problème d’ordonnancement Job-Shop **Amaro et al. (2022a)**, la résolution des problèmes de planification **Stollenwerk et al. (2020)**; **Pires et al. (2021)**, La résolution du TSP **Radzihovsky et al. (2019)**, les problèmes de coloration de graphe et le problème du sac à dos **van Dam et al. (2021)**. La majorité de ces applications ont limité leurs tests à de petites instances à cause des limites du nombre de qubits disponibles sur les ordinateurs quantiques et sur les simulateurs. Il y a des travaux qui ont orienté leurs recherche vers comment encoder les POCs d’une manière à optimiser le nombre de qubits nécessaire pour exécuter l’algorithme **Tabi et al. (2020)**. La qualité des résultats obtenue dépend du paramètre  $p$ , les tests ont montré que plus le nombre de  $p$  est grand plus la qualité des résultats obtenus est meilleure. La performance de QAOA peut s’améliorer d’une manière remarquable en augmentant  $p$  comme le montre la figure 2.9.

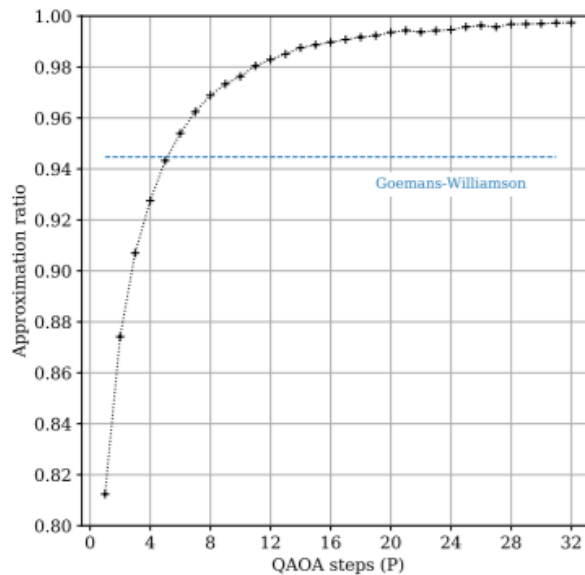


FIGURE 2.8 – La qualité des résultats de test de QAOA pour la résolution des problèmes de maxcut avec 10 noeuds, tiré de **Cook et al. (2020)**

La figure 2.9 montre qu’en augmentant le nombre d’itérations  $p$ , les résultats obtenus deviennent meilleurs. Pour  $p = 5$  le ratio obtenu est meilleur par rapport à l’algorithme classique Goemans-Williamson **Cook et al. (2020)**

Afin de mettre au point cette méthode, plusieurs propositions ont été suggérées. En 2020 **Egger et al. (2021)** ont proposé une version améliorée de QAOA qui utilise une solution initiale générée par un algorithme classique au lieu de commencer l’algorithme par l’état  $|+\rangle$  **Egger et al. (2021)**. Une autre version de QAOA **Rieffel et al. (2020)** utilise un opérateur unitaire mixer différent afin de pouvoir améliorer la qualité de la solution.

### 2.4.3 Méthodes de résolution des POCs inspirés quantiques sur un ordinateur classique

Afin de tirer profit des avantages du calcul quantique et d'améliorer les méthodes de résolution des POCs, un nouveau type d'algorithme est apparu : les algorithmes inspirés quantiques. Ces méthodes combinent les méthodes classiques avec les concepts de calcul quantique tel que la superposition et les opérateurs quantiques. Ils existaient même avant l'apparition des ordinateurs quantiques et leurs exécutions ne nécessitent pas la présence des machines quantiques **Ross (2019)**. Dans ce qui suit nous allons présenter les méthodes inspirées quantiques les plus connues.

En 1996, **Narayanan & Moore (1996)** ont proposé l'idée d'utiliser les principes du calcul quantique avec les méthodes évolutionnaires, plus précisément avec les algorithmes génétiques pour la résolution du TSP. La principale différence entre ces algorithmes et les algorithmes évolutionnaires est que les algorithmes évolutionnaires quantiques (En anglais Quantum evolutionay algorithms QEA ) manipulent des populations quantiques et utilisent le principe de superposition pour avoir une population plus importante. Par rapport aux méthodes évolutionnaires classiques, ces méthodes offrent une meilleure diversité, une recherche globale plus performante ainsi qu'une convergence rapide **Li et al. (2020)**. Dans ce qui suit nous présentons deux types de ces algorithmes :

#### A. Les algorithmes génétiques inspirés quantiques

Tout comme les algorithmes génétiques, les algorithmes génétiques quantiques (En anglais Quantum genetic algorithms QGA) manipulent une population de chromosomes. La particularité de ces chromosomes réside dans le fait qu'ils sont des chromosomes quantiques en superposition de plusieurs états. Cette population est appelée la population quantique  $Q(t) = [q_t^1, q_t^2, \dots, q_t^n]$  où  $t$  représente l'itération de l'algorithme et  $n$  la taille de la population. Un chromosome de la génération est représenté comme suit :

$$q_j^t = \begin{bmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \dots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \dots & \beta_{jm}^t \end{bmatrix}$$

tel que  $m$  représente le nombre de qubits nécessaire pour représenter un individu et  $\alpha$  et  $\beta$  représentent les amplitudes de probabilité d'être dans l'état 0 et 1 respectivement. Afin de générer la nouvelle population, ces algorithmes utilisent des opérateurs quantiques tels que les opérateurs de rotation **Ross (2019)**.

---

**Algorithme 1** : Algorithmes génétique inspiré quantique

---

```
1 procedure QGA
2 Générer population quantique initiale  $Q(t)$ 
3 Construire  $P(t)$  en observant les individus de  $Q(t)$  tel que La population  $P(t)$ 
   comporte les résultats des mesures des résultats quantiques.
4 Évaluer les individus de  $P(t)$ 
5 Stocker la meilleure solution  $b$  de  $P(t)$ 
6 while le critère d'arrêt n'est pas atteint do
7    $t \leftarrow t + 1$ 
8   Mettre à jour la population  $Q(t)$  :  $Q(t) \leftarrow U.Q(t)$ 
9   Evaluer  $P(t)$ .
10  Mettre à jour la meilleure solution  $b$  de  $P(t)$ .
11 end
12 end procedure
```

---

**B. Les algorithmes évolutionnaires inspirés quantiques**

Cet algorithme représente une évolution de l'algorithme génétique QEA. Il exploite une population quantique tout comme l'algorithme précédent, la différence la plus notable entre les deux est le concept de migration introduit dans les algorithmes évolutionnaires quantiques **Han & Kim (2002)**. Ce concept permet d'induire une variation des probabilités d'un chromosome quantique **Ross (2019)**. La migration est un paramètre de conception défini comme le processus de copier  $B$  vers  $B(t)$  tel que  $B$  représente la meilleure solution et  $B(t)$  l'ensemble de toutes les solutions associées à l'itération  $t$ . Une migration globale est mise en œuvre en remplaçant toutes les solutions de  $B(t)$  par  $B$ . Alors qu'une migration locale est mise en œuvre en remplaçant certaines des solutions de  $B(t)$  par la meilleure d'entre elles **Ross (2019)**. L'amélioration des méthodes évolutionnaires est basée principalement sur le changement d'opérateurs  $U$  utilisé pour la génération de la population quantique ou par l'intégration des opérations classiques de croisement et de mutation pour avoir plus de diversité et d'intensification.

---

**Algorithme 2** : Algorithme évolutionnaire inspiré quantique

---

```
1 procédure QEA
2 Générer population quantique initiale  $Q(t)$ 
3 Construire  $P(t)$  en observant les individus de  $Q(t)$ 
4 Évaluer les individus de  $P(t)$ 
5 Stocker les meilleures solutions de  $P(t)$  dans  $B(t)$ .
6 while le critère d'arrêt n'est pas atteint do
7    $t \leftarrow t + 1$ 
8   Mettre à jour la population  $Q(t)$  :  $Q(t) \leftarrow U.Q(t)$ 
9   Evaluer  $P(t)$ .
10  Stocker les meilleures solutions de  $B(t-1)$  et  $P(t)$  dans  $B(t)$ .
11  Stocker la meilleure solution  $b$  de  $B(t)$ .
12  if Si condition de migration then
13    | Migration global ou local.
14  end
15 end
16 end procédure
```

---

les méthodes évolutionnaires ont été largement utilisées pour la résolution d'une large gamme des POCs contrairement aux algorithmes présentée auparavant. En plus, grâce aux performances des ordinateurs classiques, ils ont été testés sur des instances assez grandes et de tirer plus de diversité grâce à l'utilisation des concepts du calcul quantique Ross (2019).

## 2.5 Les méthodes de résolution quantique proposés pour la résolution des problèmes d'ordonnancements

Les problèmes d'ordonnancement sont considérés parmi les problèmes les plus abordés par la recherche opérationnelle. L'importance de ces problèmes réside dans le fait qu'ils soient présents dans de nombreux domaines comme l'économie, l'industrie, le transport. . . , etc. Le fait de trouver une planification optimale apporte différents avantages tels que la satisfaction des clients et la réduction des coûts et des délais.

### 2.5.1 Éléments de base d'un problème d'ordonnancement

Un problème d'ordonnancement se définit par l'allocation des ressources disponibles pour des tâches sur une période donnée dans le but de satisfaire un ou plusieurs objectifs. ces objectifs peuvent être la recherche d'un ordonnancement qui minimise le temps d'exécution de la dernière tâche sur la dernière machine (makespan), le temps d'attente où

même le retard d'exécution des tâches **Zouambi (2019)**. Parmi les définitions proposées pour définir un problème d'ordonnancement dans la littérature, nous citons la définition de (Esquirol et Lopez, 1999) : " *Le problème d'ordonnancement consiste à organiser dans le temps la réalisation de **tâches**, en tenant compte des **contraintes temporelles** (délais, contraintes d'enchaînement...) et de **contraintes** portant sur l'utilisation et la disponibilité des **ressources** requises.* ".

Un problème d'ordonnancement se caractérise par quatre éléments principaux à savoir les tâches, les ressources, les contraintes et les objectifs.

1. **Les tâches** peuvent être définies comme un travail élémentaire où un ensemble d'opérations dont la réalisation nécessite un certain nombre d'unités de temps (sa durée) et d'unités de ressources. Certaines tâches peuvent être préemptives, c'est-à-dire peuvent être exécuté par morceaux. Chaque tâche  $j$  est caractérisée par :

- (a)  $p_{i,j}$  : le temps de traitement de la tâche  $j$  sur la machine  $i$  (Processing time).
- (b)  $r_{i,j}$  : la date de lancement de la tâche  $j$ , appelée aussi date de disponibilité. Il s'agit de l'heure à laquelle la tâche arrive dans le système, c'est-à-dire l'heure la plus proche à laquelle la tâche  $j$  peut commencer à être traitée (Release date).
- (c)  $c_{i,j}$  : le temps de fin d'exécution (traitement) de la tâche  $i$  sur la machine  $j$  (Completion time).
- (d)  $t_{i,j}$  : le temps de début d'exécution de la tâche  $i$  sur la machine  $j$  (Starting time).
- (e)  $d_{i,j}$  : la date d'échéance de la tâche  $j$  qui représente sur la machine  $j$  la date d'expédition où d'achèvement promise (c'est-à-dire la date promise au client) (Due date).

2. **Les contraintes** sont les règles qui doivent être respectées lors de la création d'un planning pour qu'il soit réalisable. Les contraintes définissent la plage de valeurs que les variables de décision, qui sont liées aux tâches et aux ressources **Carlier et al. (1993)**. Nous pouvons distinguer deux types de contraintes :

- (a) Les contraintes de ressources qui représentent le fait que les ressources sont disponibles en quantité limitée sur une période régulière. La capacité limitée d'une ressource implique un nombre particulier de tâches qui ne doivent pas être dépassées sur cette ressource. Ces contraintes peuvent être disjonctives causant des contraintes sur la réalisation des tâches sur la même ressource dans une période où cumulatives, engendrant des limites sur le nombre de tâches à réaliser en parallèle.
- (b) Les contraintes temporelles qui engendrent les contraintes liées aux valeurs que les variables temporelles peuvent prendre. Nous citons les contraintes temporelles, qui sont liées aux délais des tâches (délais de livraison, disponibilité des

approvisionnements) ou à la durée globale d'un projet, des contraintes de précédence qui définissent comment certaines tâches sont positionnées par rapport à d'autres et les contraintes de dates au plus tôt, dues à la non-disponibilité de certains paramètres requis pour commencer l'exécution d'une tâche

3. **Les ressources.** la ressource peut être définie comme le moyen, la technique où le facteur humain, dont la contrainte de la disponibilité limitée ou non, les ressources sont destinées à être utilisé pour la réalisation d'une où plusieurs tâches **Carlier et al. (1993)**. Deux types de ressources peuvent être distingués :

- (a) Ressources renouvelable qui peuvent redevenir à nouveau disponible après être allouées à une tâche. Nous pouvons distinguer les ressources partageables (commutatives) qui peuvent être utilisées par un nombre illimité de tâches, et les ressources non partageables (disjonctives) qui ne peuvent pas exécuter plusieurs tâches simultanément.
- (b) Ressources consommables si après avoir été allouées à une tâche ou ensemble de tâches, ne peuvent redevenir disponible comme l'argent et la matière première.

4. **Les critères d'optimalité.** Afin d'évaluer la qualité d'un ordonnancement, un ou plusieurs objectifs peuvent être visé. Ces derniers peuvent varier selon le problème traité parmi lesquelles nous pouvons citer :

- (a) La durée totale d'un ordonnancement (makespan) qui consiste à minimiser la durée totale d'accomplissement des jobs ( $C_{max}$ ) qui représente le max entre les temps de la fin d'exécution des tâches.
- (b) Le temps de latence qui sert à minimiser le temps de retard algébrique défini par l'équation  $L_{max} = \max(L_{j,i})$  tel que  $L_j = C_{j,i} + d_{j,i}$ .
- (c) Temps d'achèvement pondéré total qui représente la somme pondérée des temps d'achèvement.

### 2.5.2 Notation des problèmes d'ordonnancement

Afin de pouvoir distinguer entre les différents problèmes d'ordonnancement, nous adoptons la notation  $\alpha|\beta|\gamma$  **Graham et al. (1979); Blazewicz et al. (1983)**. Cette notation se caractérise par les trois champs  $\alpha$  pour les ressources,  $\beta$  pour les tâches et  $\gamma$  les critères à optimiser.

Le champ  $\alpha$  revient à l'organisation de l'atelier et les ressources. Ce champ est composé de deux parties  $\alpha = \alpha_1\alpha_2$  telles que  $\alpha_1$  représente le type d'atelier et  $\alpha_2$  décrit le nombre de machines. Le champ  $\alpha_1$  peut prendre les valeurs de l'ensemble  $\{\emptyset, P, Q, R, F, J, O\}$  tel que :

- $\emptyset$  pour les problèmes à machine unique.
- P pour les problèmes à machines parallèles et identiques.

- Q pour les problèmes à machines parallèles et uniformes.
- R pour les problèmes à machines parallèles et indépendants.
- F pour les problèmes d'atelier de type Flowshop.
- J pour les problèmes d'atelier de type Jobshop.
- O pour les problèmes d'atelier de type Openshop

Le deuxième champ  $\beta$  représente les caractéristiques des tâches et des machines. il décrit les éléments qui doivent être pris en considération, parmi lesquelles :

- *pmtn* si la préemption est autorisée.
- $p_{ij}$  si toutes les durées opératoires sont égales à une valeur  $p$ .
- $r_i$  si chaque tâche  $J_i$  possède une date de disponibilité.
- $S_{ij}$  si le temps de préparation dépendant de la séquence entre les tâches  $J_i$  et  $J_j$ .
- $S_{ri}$  si le temps de préparation de la machine  $M_r$  pour la tâche  $J_i$ .
- *pmu* si l'ordre (ou permutation) selon lequel les tâches passent sur la première machine est maintenu à travers le système.
- *block* si la tâche complétée doit rester sur la machine en amont, prévenant ou bloquant

Le dernier champ  $\gamma$  représente le critère d'optimalité que les ordonnancements doivent satisfaire. Il peut être :

- $C_{max} = \max\{C_i, i = 1, \dots, n\}$  : date de fin de toutes les tâches.
- $\sum_i C_i$  : la somme des dates de fin des opérations.
- $L_{max} = \max\{L_i, i = 1, \dots, n\}$  : le retard algébrique maximum.
- $\sum_i T_i$  : le nombre de tâches en retard.
- $T_{max} = \max\{T_i, i = 1, \dots, n\}$  : le retard maximum.

### 2.5.3 Synthèse des travaux basés quantique proposés dans la littérature pour la résolution des problèmes d'ordonnancement

Dans ce qui suit, nous présentons une synthèse de l'état de l'art sur les méthodes quantiques proposés pour la résolution des problèmes d'ordonnancement.

#### — L'algorithme de Grover

**Scherer *et al.* (2021)** ont proposé une méthode quantique basée sur l'algorithme de Grover pour la résolution du problème de l'organisation des opérateurs sur un intervalle de temps et des positions dans un système aérospatial sous plusieurs contraintes. Ils ont proposé un encodage des ordonnancements ainsi qu'un oracle permettant de distinguer les ordonnancements satisfaisant les contraintes.

**Lu & Marinescu (2007)**, a proposé une méthode pour la résolution d'un problème d'ordonnancement dont le but est de trouver un ordonnancement avec le temps d'achèvement le plus court dans un environnement de machines parallèles non liées  $R||C_{max}$ . L'auteur a proposé une généralisation de l'algorithme de Grover pour la résolution de  $R||C_{max}$ .

— **Le recuit quantique**

**Venturelli et al. (2016)** ont proposé une modélisation mathématique sous la forme QUBO pour la résolution du problème du Job-Shop basé sur la modélisation "time-index" ainsi que des techniques pour la réduction du nombre de variables décisionnelles. le recuit simulé a été utilisée pour la résolution du modèle mathématique proposée.

**Goh et al. (2020)** ont proposé une méthode pour la résolution des problèmes de permutation basée sur la classification des instances et les solveurs quantiques. Ils ont testé leur méthode sur le problème du Flow-Shop après avoir transformé le Flow-Shop en un problème du TSP.

**Ikedo et al. (2019)** ont modélisé les différentes contraintes (Hard, Soft) vers des contraintes quadratiques afin d'obtenir un modèle QUBO et de pouvoir résoudre le problème d'ordonnancement des infirmières.

**Shimada et al. (2021)** ont proposé la décomposition du problème du Job-Shop en deux sous problèmes (master et sub-problèmes) à savoir le problème de l'estimation du makespan et le problème de satisfaction des contraintes. Ces problèmes sont résolus de manière répétée en utilisant une méthode classique pour l'estimation du makespan et le recuit quantique pour trouver les ordonnancements satisfaisants les contraintes.

**Zhang et al. (2022)** ont proposé un modèle QUBO pour le problème du Job-Shop basé sur le modèle disjonctif puis une décomposition du problème en deux parties.

— **Les algorithmes variationnels**

La résolution de problème de Job-Shop avec des temps d'exécution égaux pour toutes les tâches et avec la supposition qu'il y aura une seule idle time seulement au début de l'exécution. **Amaro et al. (2022a)** ont proposé une modélisation QUBO et Cvar pour pouvoir résoudre les problèmes avec les méthodes variationnel QAOA, VQE et Filtring-VQE. Les résultats obtenus montrent que la méthode Filtring-VQE donne les meilleurs résultats.

**Pires et al. (2021)** ont proposé l'utilisation de l'algorithme d'optimisation approximative quantique (QAOA) comme heuristique pour résoudre le problème de l'emploi du temps des écoles (Timetabling). Les résultats obtenus ont montré qu'il est possible d'appliquer le QAOA au problème de l'emploi du temps des écoles.



## 2.6 Synthèse des méthodes de résolution quantique proposés pour la résolution des problèmes d'optimisation combinatoire

FIGURE 2.9 – présente le timeline d'apparition des premières méthodes de résolution des POCs quantique

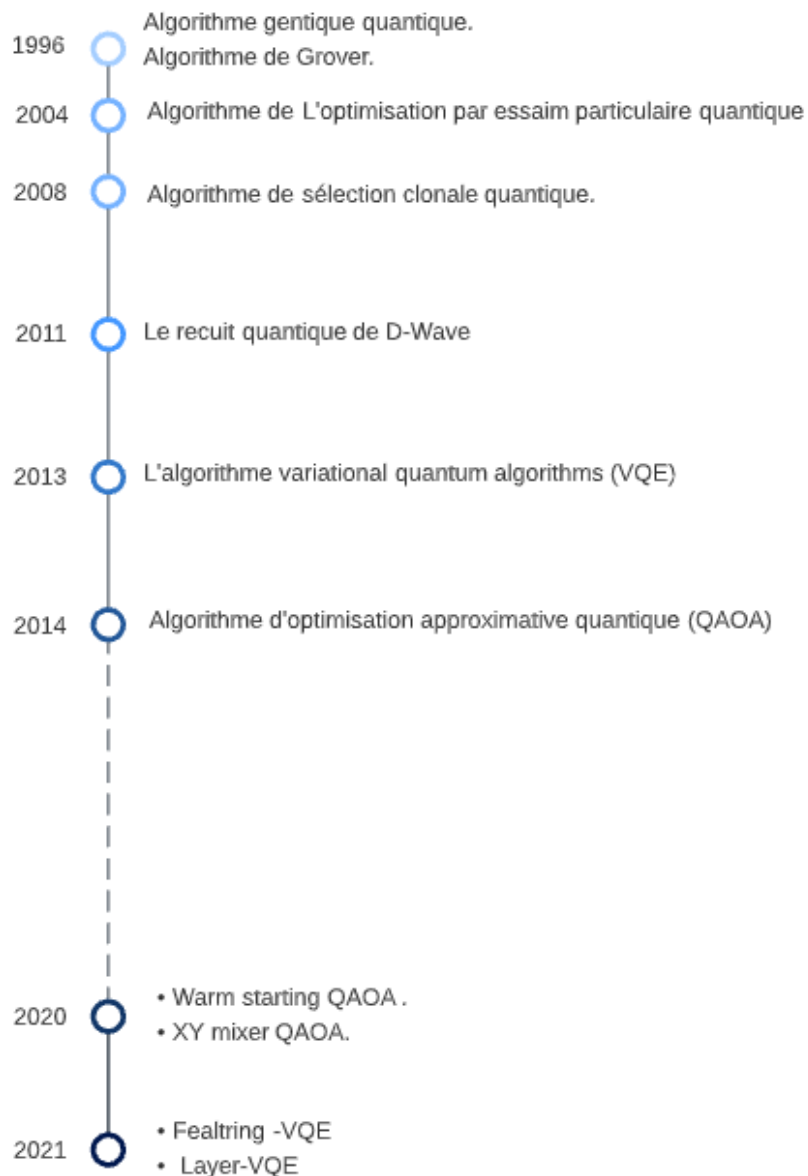


TABLE 2.2 – Récapitulatif des travaux sur la résolution des problèmes d’optimisation combinatoire avec l’algorithme de Grover

type de POC	Référence	Contribution
Le problème de maximum clique	Haverly & López (2021)	Adaptation de l’oracle de Grover pour la distinction des solutions faisables et le calcul de la fonction de coût pour la résolution de problème de max clique.
Le problème du voyageur de commerce	Srinivasan <i>et al.</i> (2018)	Résolution du problème de TSP en se basant sur l’algorithme quantique d’estimation de phase pour le calcul des coûts et l’algorithme de Grover pour trouver le meilleur chemin.
La résolution des problèmes QUBO et CPBO	Gilliam <i>et al.</i> (2021)	Une méthode basée sur GAS pour la résolution des problèmes QUBO et CPBO en utilisant les concepts des dictionnaires quantique pour réaliser l’oracle..
le problème de Coloration de la liste	Mukherjee (2022)	Proposition d’une méthode pour l’initialisation de l’espace de recherche avec seulement les solutions possibles au lieu d’utiliser l’opérateur Hadamard, ce qui permet d’explorer un espace plus restreint.
$R  C_{max}$	Lu & Marinescu (2007)	Exploration de la généralisation de l’algorithme de Grover “amplification de l’amplitude”, afin de modifier l’espace de recherche vers un espace d’état contenant la solution et le coût associés.
Les problèmes de satisfiabilité	Gilliam <i>et al.</i> (2021)	Adaptation de l’algorithme de Grover pour la résolution des problèmes de satisfiabilité.
/	Panchi & Shiyong (2008)	Modification de l’algorithme de Grover de tel sort à distinguer les solutions par leur degré d’importance.
Le problème d’ordonnement des satellites	Scherer <i>et al.</i> (2021)	Encodage de problèmes d’ordonnement étudier d’une manière permettant la réduction de nombre de qubits.
Coloration de graph	Ardelean & Udrescu (2022)	Modélisation d’un algorithme génétique réduit avec l’algorithme de Grover.

TABLE 2.3 – Récapitulatif des travaux sur la résolution des problèmes d’optimisation combinatoire avec des méthodes Hybrides

Méthode	Référence	Contribution
Warm Starting QAOA	Egger <i>et al.</i> (2021)	Modification dans l’algorithme QAOA de telle sorte qu’il utilise comme état initial une solution obtenu par un algorithme classique d’optimisation l’hamiltonien MIXER change.
XY-mixers QAOA	Rieffel <i>et al.</i> (2020)	Modification de l’Hamiltonien Mixer par défaut avec un XY mixer afin d’améliorer la qualité de la solution obtenue.
XY-mixers QAOA	Beaulieu & Pham (2021)	Résolution de problème de Max-Cut clusteringUtilisation de la méthode de Warm Starting QAOA.
VQA	Tabi <i>et al.</i> (2020)	Résolution du problème de coloration du graphe avec un encodage efficace du problème afin de réduire le nombre de qubits nécessaire.
F-VQA	Amaro <i>et al.</i> (2022b)	Proposition d’un algorithme qui utilise des opérateurs de filtrage pour parvenir à une convergence plus rapide et plus fiable vers la solution optimale et les cônes de causalité pour réduire le nombre de qubits requis sur un ordinateur quantique.
F-VQA	Amaro <i>et al.</i> (2022a)	Résolution du problème de JOB SHOP avec l’algorithme de VQE.
Layer VQE	Liu <i>et al.</i> (2022)	Proposition d’un algorithme qui utilise plusieurs couches de VQE.
CVaR-QAOA CVaR-VQE	Barkoutsos <i>et al.</i> (2020)	Amélioration du temps de convergence des méthodes QAOA et VQA avec une fonction d’agrégation qui s’appelle Cvar.
VQA	Bonet-Monroig <i>et al.</i> (2021)	Comparaison des performances des méthodes d’optimisation classique sur les algorithmes quantiques variationnels.

En visualisant le timeline 2.9, nous pouvons constater que l’amorce de l’optimisation combinatoire quantique remonte aux années 90. Après l’apparition de l’algorithme de Gro-

ver, plusieurs types d'algorithmes ont été proposés en commençant par les algorithmes de résolution des POCs inspirés quantique qui combinent les principes du calcul quantique avec les méthodes classiques de résolution des POCs. En 2007, l'entreprise D-wave a réalisé un ordinateur quantique pour la résolution des POCs avec la méthode du recuit quantique. Depuis 2013, un grand nombre de travaux ont orienté leurs recherche vers les méthodes variationnelles, car ses méthodes sont considérées comme les méthodes les plus adaptés pour les dispositifs NISQ qui comportent un taux d'erreur important, contrairement à l'algorithme de Grover qui pour le moment reste une méthode théorique et ne peut apporter des résultats importants dans la pratique à cause de sa nature.

Le tableau 2.2, présente la majorité des travaux, basée sur l'algorithme de recherche de Grover. Plusieurs améliorations ont été proposées soit en termes de nombres de qubits requis pour l'encodage des problèmes étudiés, soit en termes d'espace de recherche généré. En effet, il existe deux approches pour la génération de l'espace de recherche :

(1) l'approche utilisant la méthode d'initialisation générale de Grover avec l'opérateur Hadamard, qui implique la génération de tous les états possibles dans l'espace Hilbert exploré. Par conséquent, l'espace des états généré va contenir pour certain cas des solutions infaisables. Pour surmonter cette limite, nombreux travaux ont suggéré d'adapter la partie de l'oracle pour qu'il puisse distinguer les solutions faisables et non faisables.

(2) La deuxième approche génère un espace de recherche contenant seulement les solutions possibles en modifiant l'opérateur utilisé pour l'initialisation des états et le circuit de diffusion selon le problème traité.

Dernièrement, des chercheurs ont émis l'idée d'un algorithme de Grover modifié pour distinguer les solutions par leurs degrés d'importance. Par conséquent, les amplitudes associées aux solutions seront liées au degré d'importance de ces dernières. Contrairement à l'algorithme de Grover classique qui augmentent les amplitudes de toutes les solutions trouvées avec la même valeur. Cette méthode s'avère très intéressante, surtout si nous pouvons l'adapter pour la résolution des POCs.

Le tableau 2.3, présente les différents travaux dont le but principal était l'amélioration des méthodes variationnelles. Les auteurs de ces travaux ont proposé des solutions pour surmonter certaines limites tel que la convergence vers des solutions qui ne sont pas valides, la complexité du circuit en termes de nombre de qubits et profondeur et le temps de convergence. La méthode la plus récente est F-VQE proposée en 2021 utilisée seulement pour la résolution des problèmes de coupe maximale et Job-Shop. Dans les deux travaux, cette méthode a montré ces performances par rapport à VQE et QAOA. Les résultats ont montré que la méthode VQE présente les meilleurs résultats par rapport à la méthode QAOA.

## 2.7 Conclusion

Dans ce chapitre, nous avons essayé de présenter une nouvelle voie de recherche qui combine deux domaines très puissants, l'optimisation combinatoire et le calcul quantique. Nous avons essayé d'une part de présenter le calcul quantique, la théorie de l'information quantique ainsi que l'informatique quantique. D'autre part, nous avons présenté les différentes méthodes quantiques utilisées pour la résolution des problèmes d'optimisation combinatoires.

Nous avons présenté une classification des méthodes de résolution des POCs quantiques, puis nous avons orienté notre recherche bibliographique vers deux types de méthodes : l'optimisation globale avec l'algorithme de Grover et les algorithmes variationnels. Cette piste semble très intéressante à explorer, surtout vu qu'elle reste encore nouvelle et très peu de travaux ont été publiés dans ce cadre.

Notre recherche bibliographique nous a permis de nous positionner dans le contexte courant de la recherche et de pouvoir définir les challenges du domaine. Bien que l'algorithme de Grover propose une amélioration importante par rapport aux méthodes classiques, l'utilisation de cet algorithme était restreinte à un petit nombre de POCs. D'un autre côté, les algorithmes variationnels qui représentent des méthodes prometteuses, surtout avec l'arrivée de la nouvelle génération des ordinateurs quantiques qui vont comporter un nombre de qubits assez grand et un taux de bruits important. Ces méthodes ont connu un progrès très important ces dernières années, et plusieurs améliorations ont été présentées, mais l'étude de leur performance reste encore une question ouverte à cause du nombre restreint des travaux publiés dans les domaines. Le troisième type d'algorithme qui combine le calcul quantique avec les algorithmes de résolution des POCs semble le seul type qui peut être utilisé en pratique pour la résolution des grandes instances.

Nous avons constaté que les problèmes d'ordonnancement ont été rarement traités, notamment, le problème du Flow-Shop qui n'a jamais été résolu auparavant ni avec l'algorithme de Grover ni avec les algorithmes variationnels. Ceci nous encourage à orienter notre sujet de PFE vers la résolution de ce problème par ces méthodes. Un tel travail représente une bonne contribution dans le domaine non seulement pour pouvoir résoudre le problème du Flow-Shop avec des nouvelles approches, mais aussi pour étudier ces méthodes.

# Chapitre 3

## Approche quantique basée sur l'algorithme de Grover

### 3.1 Introduction

Dans le chapitre précédent, nous avons présenté les problèmes d'ordonnancement ainsi qu'une synthèse des travaux proposant des méthodes quantiques pour la résolution de ce type de problème. Nous avons également constaté que le problème du Flow-Shop n'a jamais été résolu avec l'algorithme de Grover et les méthodes hybrides. Par conséquent, nous avons défini comme objectif de notre mémoire la résolution du problème du Flow Shop de permutation (PFSP) avec les méthodes quantiques basées sur l'algorithme de Grover et les algorithmes variationnels.

Dans ce chapitre, nous présentons notre première méthode Grover-FSP basée sur l'algorithme de Grover. Dans un premier lieu, nous allons commencer par présenter l'architecture générale de l'approche proposée. La deuxième section est dédiée à l'explication du circuit de Grover proposé et plus précisément, nous allons détailler comment nous avons adapté la partie oracle de Grover pour la résolution de PFSP. Enfin, nous présenterons les deux méthodes qui permettent d'exploiter le circuit de Grover pour l'optimisation globale.

### 3.2 Contexte de l'étude - Le Flow-Shop de permutation

Un problème d'ordonnancement de type Flow-Shop se compose de deux éléments principaux : (1) un système de production composé de  $m$  machines notées  $M_1, M_2, \dots, M_m$  ; et (2) un ensemble de  $n$  tâches notées  $N_1, N_2, \dots, N_n$  à traiter sur ces machines. Toutes les tâches ont le même ordre de traitement sur les  $M$  machines. Chaque tâche  $N_i$  se compose de  $m$  opérations  $O_{i1}, O_{i2}, \dots, O_{im}$  et chaque opération  $O_{ij}$  possède un temps d'exécution sur une machine  $j$  ( $P_{i,j}$ ).

Les problèmes d'ateliers Flow-Shop peuvent être classés en deux grandes catégories : les

cycliques et les non-cycliques. À son tour, chacune de ces deux catégories se décompose en problème no-wait, blocking, et buffer limité et illimité. Une autre classification des problèmes d'ordonnancement de type Flow-Shop peut être faite en fonction de la logique de traitement des tâches. Chacune de ces catégories est décrite brièvement ci-dessous **Bagchi et al. (2006)**.

- Non-cyclique. la plupart des recherches antérieures sur les problèmes d'ordonnancement d'ateliers de fabrication portent sur l'ordonnancement non cyclique, dans lequel un ensemble de tâches doit être ordonné sur un horizon de planification afin d'optimiser certaines mesures de performance.
- Cyclique. Avec l'augmentation de l'utilisation des systèmes de production en flux continu qui exigent que les produits de plusieurs types de tâches soient fabriqués en fonction de leur demande, les mesures classiques (par ex,  $C_{max}$  ) ne sont plus applicables. Par conséquent, l'orientation de la fabrication moderne s'est orientée vers l'ordonnancement cyclique, où le plus petit ensemble de tâche qui satisfait le taux de demande respectif des divers produits est produit de manière cyclique **Wittrock (1985)**.
- No-wait. Dans ces ateliers, chaque travail doit être traité du début à la fin, sans aucune interruption sur ou entre les machines.
- Blocking. Dans ces ateliers, il n'y a pas de buffer entre les machines. Les travaux partiellement traités ne peuvent pas quitter la machine sur laquelle ils sont traités, sauf si une machine est disponible en aval.
- Unlimited-buffer. Dans ces ateliers, un nombre illimité de travaux partiellement exécutés peuvent rester temporairement dans des buffers entre les machines.
- Limited-buffer. Dans ces ateliers, un nombre limité de travaux partiellement finis peuvent temporairement rester dans les buffers entre les machines.

Dans ce travail, nous nous intéressons à l'étude d'un problème d'ordonnancement de type "Flow-Shop de permutation" avec  $n$  tâches, composée de  $M$  opérations. Toutes les tâches doivent être traitées de manière non-préemptive sur  $m$  machines dans le même ordre. La capacité de l'espace intermédiaire est infinie et une machine ne peut traiter qu'une seule opération à la fois. Notre but est la minimisation du temps de complétion de la dernière tâche sur la dernière machine notée  $C_{max}$ . La fonction objectif est exprimé comme suit :

$$\text{Minimiser } \text{Max } C_{i,j}$$

$$C_{i,j} = S_{i,j} + P_{i,j} \tag{3.1}$$

$$S_{i,j} = \max\{C_{i-1,j}, C_{i,j-1}\} \text{ tel que } S_{0,0} = 0 \tag{3.2}$$

tel que :

- $S_{i,j}$  représente le temps du début de la  $i$ -ème opération du  $j$ -ème job de la séquence ;
- $C_{i,j}$  représente le temps d'achèvement de la  $i$ -ème opération du  $j$ -ème job de la séquence ;
- $P_{i,j}$  représente le temps du traitement de la  $i$ -ème opération du  $j$ -ème job dans la séquence.

### 3.3 Architecture générale de l'approche

Dans cette partie, nous allons présenter l'architecture générale de l'approche utilisée pour la résolution du problème de Flow-Shop. L'idée repose essentiellement sur l'utilisation de l'algorithme de Grover comme un sous-algorithme pour trouver les solutions réalisables qui ont un  $C_{max}$  inférieur à la borne supérieure. Dans ce qui suit, nous allons introduire les trois étapes de l'approche.

La première étape consiste en l'estimation d'une borne supérieure en utilisant des heuristiques classiques tel que les heuristiques NEH **Nawaz *et al.* (1983)** et CDS **Campbell *et al.* (1970)** où aléatoirement. Cette borne sert comme une entrée au circuit du Grover qui cherche à trouver toutes les solutions possibles ayant un  $c_{max}$  inférieur à cette borne supérieure. Aussi, nous devons définir le nombre de rotations de l'algorithme de Grover.

La deuxième étape consiste en la préparation des états quantiques et l'encodage des variables sous forme de vecteurs quantiques. Après l'encodage des variables et l'estimation de la borne supérieure, l'étape prochaine est la construction du circuit de Grover qui comporte deux parties très importantes : l'oracle et l'opérateur de diffusion. l'oracle permet de marquer toutes les solutions faisables ayant un  $c_{max}$  inférieur à la borne supérieure et le circuit de diffusion permet d'amplifier la probabilité d'apparition des solutions marquées par l'oracle. Le circuit de Grover consiste à itérer les deux opérateurs : oracle et diffuseur plusieurs fois appelées **rotations**.



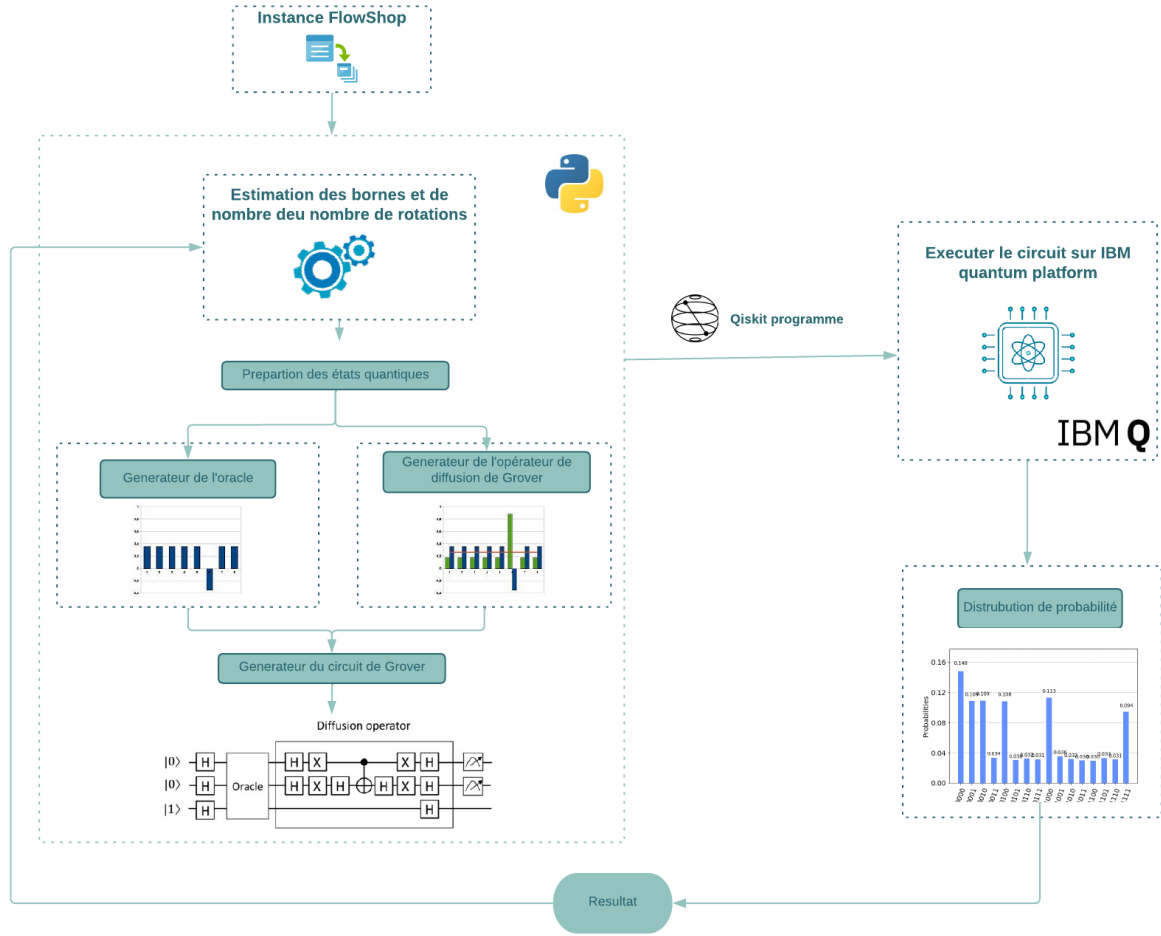


FIGURE 3.1 – Schéma général présentant l'approche Grover-FSP

La troisième étape consiste en l'exécution du circuit de Grover. Nous avons opté pour l'utilisation des simulateurs quantiques à cause des limites des ordinateurs quantiques disponibles aujourd'hui et nous avons utilisé la plateforme IBM quantum qui offre plusieurs types de simulateurs quantiques et nous permet de surmonter la limite du nombre de qubits qu'on peut simuler localement. Après l'exécution du circuit, nous allons obtenir une distribution de probabilité des possibilités d'apparition de toutes les solutions possibles. Nous choisissons une des solutions trouvées comme nouvelle borne supérieure et nous répétons le même processus jusqu'à atteindre un critère de fin défini auparavant.

### 3.4 Phase 1 : Encodage de la solution

Dans cette section, nous allons présenter la représentation des différents variables et données sous forme de vecteurs quantiques. Tout d'abord, nous commençons par représenter la variable job  $J_i$  et la variable de temps de fin d'exécution,  $C_{i,j}$  après la représentation des permutations ainsi que la génération de l'espace de recherche. Dans ce qui suit, nous allons considérer une instance du FlowShop avec  $N = 2^n$  et  $M = 2^m$  qui représentent

respectivement le nombre de tâches et le nombre de machines. Chaque job  $i$  doit passer par toutes les machines et possèdent un temps d'exécution  $P_{ij}$  sur chaque machine  $j$  et un temps de fin d'exécution sur chaque machine  $C_{ij}$ .

- **- Job-Position.** Comme première étape, nous allons commencer par la représentation d'une tâche comme un vecteur quantique ket de  $n$  qubits ou la projection de chaque état quantique donne la représentation binaire de la tâche. Par exemple la tâche 1 est encodée par  $|001\rangle$  sur 3 qubits. Cependant, chaque job doit être assigné à une position  $p$  ce qui implique que l'encodage des tâches sous forme des vecteurs quantiques n'est pas suffisante, car nous aurons besoin d'encoder l'information tâche  $i$  qui est affectée à une position  $k; k = 1, \dots, N$ . Pour ce faire, nous définissons le vecteur quantique tâche-position  $|X\rangle_k$  comme suit :

$$|X\rangle_k = \begin{cases} |00\dots00\rangle_k : \text{tâche 0 (00\dots00 en binaire) est affectée à la position } k \\ |00\dots01\rangle_k : \text{tâche 1 (00\dots01 en binaire) est affectée à la position } k \\ |00\dots10\rangle_k : \text{tâche 2 (00\dots10 en binaire) est affectée à la position } k \\ \dots \\ |11\dots1\rangle_k : \text{tâche } N \text{ (11\dots11 en binaire) est affectée à la position } k \end{cases}$$

Une représentation similaire a été utilisée dans **Scherer *et al.* (2021); Lu & Marinescu (2007)**.

**Exemple.** Soit un atelier composé de deux machines et 4 jobs de production et la séquence Job 1 après Job 2 après Job 0 après Job 3.

Nous définissons d'abord le nombre de qubits nécessaire pour représenter l'état quantique job-position. Comme nous avons seulement 4 jobs, nous aurons besoin d'utiliser 2 qubits pour les représenter. Par conséquent, les vecteurs job-position associés à ce dernier sont représentés comme des vecteurs dans l'espace  $\mathbb{C}^4$  comme suit :

$$|01\rangle_0 \sim \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \in \mathbb{C}^4, |10\rangle_1 \sim \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \in \mathbb{C}^4, |00\rangle_2 \sim \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \in \mathbb{C}^4, |11\rangle_3 \sim \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \in \mathbb{C}^4$$

- **- Permutation.** Notre espace de recherche est composé des différents ordonnancements possibles des jobs ou chaque ordonnancement représente une permutation entre les différents jobs d'où la nécessité de définir une méthode de génération d'une permutation donnée  $S_\sigma$ . il s'agit principalement d'appliquer le produit tensoriel entre les  $N$  vecteurs job-position :

$$|S_\sigma\rangle = \bigotimes_{k=1}^N |X\rangle_k$$

Le produit tensoriel nous permet de générer un vecteur permutation comportant l'affectation des  $N$  jobs à des positions, ce qui implique qu'une permutation est représentée par un état quantique avec  $p = n * N$  qubits tels que  $N$  et  $n$  représentent respectivement le nombre de Jobs et le nombre de bits nécessaires pour encoder un job en binaire .

**Exemple.** Considérons l'exemple présenté précédemment avec l'ordonnancement suivant :  $S_\sigma : \{Job_2, Job_3, Job_1, Job_3\}$ . Notre permutation va être égale au produit tensoriel des vecteurs job-position  $|X_2\rangle_0, |X_3\rangle_1, |X_1\rangle_2, |X_4\rangle_3$ , comme suit :

$$|S_\sigma\rangle = \bigotimes_{k=1}^N |X\rangle_k = |X_1\rangle_0 \otimes |X_2\rangle_1 \otimes |X_0\rangle_0 \otimes |X_3\rangle_0$$

$$|S_\sigma\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |01100011\rangle$$

En effet, les deux premiers qubits représentent le job (en binaire) affecté à la position 0, puis les deux qubits suivants représentent le job (en binaire) affecté à la position 1 et les deux derniers qubits représentent le job affecté à la dernière position.

01	10	00	11
position 0	position 1	position 2	position 3

Afin de pouvoir générer toutes les permutations possibles  $|S\rangle$ , nous utilisons l'opérateur quantique Hadamard qui permet d'obtenir une superposition uniforme de toutes les permutations possibles.

$$|S\rangle = \frac{1}{\sqrt{2^{n \cdot N}}} \sum_{\sigma=1}^{2^{n \cdot N}} |S_\sigma\rangle$$

Il est possible de se retrouver parfois avec des permutations non valides, et ce à cause de l'utilisation de l'opérateur Hadamard. Pour pouvoir identifier ces permutations, une étape intermédiaire pour assurer la validité du résultat obtenu est ajouté dans le circuit de Grover.

- **- Temps de complétion**  $C_{ij}$ . La dernière variable à encoder est le temps de complétion (fin d'exécution)  $C_{k,j}$  du job affecté à la position  $k$  sur la machine  $j$ . elle égal à :

$$C_{ij} = \max(C_{(i-1)j}, C_{i(j-1)}) + P_{ij}; \quad \forall i, \forall j$$

Toutes les variables  $C_{ij}$  doivent être encodé avec le même nombre de qubits. Cette variable est représenté sur  $q$  qubits tels que  $q$  est une estimation du nombre de qubits nécessaire pour encoder le plus grand  $C_{max}$  possible. Afin de trouver cette estimation, nous utilisons un borne supérieure pour le  $C_{max}$  comme suit :

$$C_{max} \leq (N + M - 1)p_{max}$$

où  $p_{max}$  représente le plus grand temps d'exécution et le nombre de qubits nécessaires dans ce cas est égal à

$$\lceil \log_2((N + M - 1)p_{max}) \rceil$$

Chaque  $C_{kj}$  est initialisé par le temps d'exécution  $P_{ij}$  de job  $i$  affecté à la position sur la machine  $j$ .

**Exemple.** Soit l'exemple précédent avec les temps d'exécution décrit dans le tableau 3.1

$p_{ij}$	$M_1$	$M_2$
$J_0$	1 (01)	2 (10)
$J_1$	2 (10)	3 (11)
$J_2$	2 (10)	3 (11)
$J_3$	2 (10)	3 (11)

TABLE 3.1 – Les temps d'exécution de 4 jobs sur les deux machines

Considérons l'ordonnancement suivant :  $J_0 \rightarrow$  position 2,  $J_1 \rightarrow$  position 4,  $J_2 \rightarrow$  position 3 and  $J_3 \rightarrow$  position 1. Le nombre des qubits nécessaire pour encoder le vecteur  $C_{ij}$  est égale à 5 qubits. Si on prend par exemple, le temps d'exécution de la tâche affectée à la position 2 sur la machine 2 est initialisée comme suit :  $|C_{2,2}\rangle = |00010\rangle$ .

### 3.5 Phase 2 : Construction du circuit de Grover

Dans cette section, nous introduisons l'algorithme de Grover adapté pour la résolution du problème du Flow Shop plus précisément pour la recherche des solutions ayant un  $C_{max}$  inférieur à une borne supérieure  $|U_{bs}\rangle$ . Dans un premier temps, nous allons commencer par présenter la partie initialisation de l'espace de recherche, puis nous allons montrer comment nous avons adapté l'opérateur "oracle" pour pouvoir distinguer les solutions de notre problème ainsi que la construction de l'opérateur de diffusion qui nous permet d'amplifier les amplitudes des solutions.

Le circuit du Grover exploite l'avantage du parallélisme quantique permettant à une fonction de traiter toutes les possibilités simultanément, ce qui offre plus de performance. L'algorithme commence par la génération d'un état quantique en superposition de toutes les permutations possibles, puis l'itération de Grover est répétée plusieurs fois afin de pouvoir obtenir à la fin une superposition comportant seulement les solutions qui satisferont les deux contraintes : une permutation qui ne contient pas une répétition des tâches et qui possède un  $C_{max}$  inférieur à une borne supérieure  $U_b$ . L'itération de Grover consiste en deux parties essentielles : l'opérateur oracle et l'opérateur de diffusion. L'opérateur oracle marque les permutations représentant une solution avec un signe négatif en inversant la phase de l'état quantique. L'opérateur de diffusion va amplifier la probabilité d'apparition des états marqués et la dernière étape représente la projection des états quantiques vers des états classiques en appliquant une mesure. Dans ce qui suit, nous allons expliquer en détails les différentes parties du circuit proposé pour la résolution du Flow-Shop.

### 3.5.1 Étape 1 : L'initialisation

La première étape de l'algorithme proposé consiste à superposer tous les états de l'espace de recherche, ce dernier comporte toutes les permutations possibles ou chaque permutation est représentée comme un état quantique avec  $p = N * n$  qubits comme décrit dans la section précédente 3.4.

Nous initialisons notre état quantique  $|perm\rangle_p$  dans l'état  $|0\rangle_p$  puis nous appliquons l'opérateur Hadamard (H) sur tout le registre  $|perm\rangle_p$  pour pouvoir générer un état quantique en superposition uniforme de toutes les permutations possibles y compris les permutations non valides. Nous pouvons décrire cette opération par l'équation suivante 3.3

$$|\psi_1\rangle = H^{\otimes p} |perm\rangle_p = \frac{1}{\sqrt{2^p}} \sum_{x=0}^{2^p-1} |x\rangle \quad (3.3)$$

Prenons l'exemple de 2 jobs donc 2 positions possibles et un seul qubit pour représenter une position. Par conséquent, le nombre de qubits pour représenter une permutation  $p$  est égal à 2 qubits et le nombre de permutations possible est égal à 4 :  $|00\rangle \sim \{job_0, job_0\}$ ,  $|01\rangle \sim \{job_0, job_1\}$ ,  $|10\rangle \sim \{job_1, job_0\}$ ,  $|11\rangle \sim \{job_1, job_1\}$ . Il est clair que les permutations  $|00\rangle$  et  $|11\rangle$  ne sont pas valides pour pouvoir les éliminer, un traitement supplémentaire va être appliqué dans la partie oracle. Le circuit permettant de générer toutes les permutations possibles est décrit dans la figure 3.8.

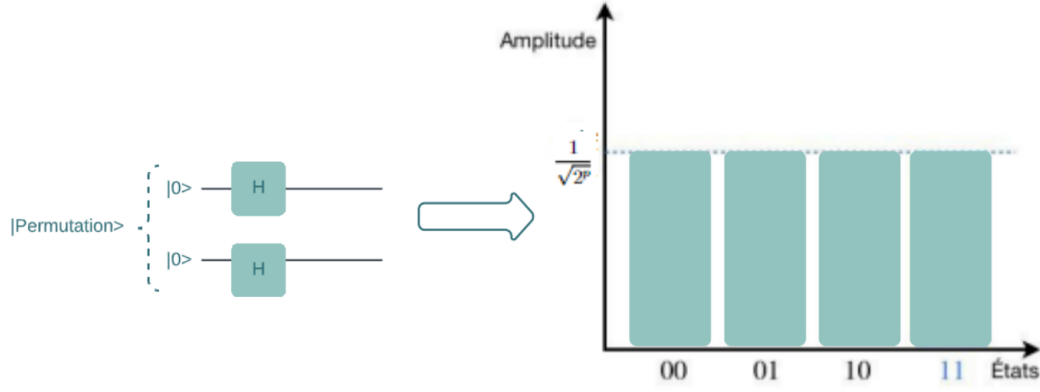


FIGURE 3.2 – Initialisation

### 3.5.2 Étape 2 : L'opérateur oracle

Le rôle principal de l'oracle est la distinction des ordonnancements valides ; qui comportent tous les tâches ; et qui possèdent un  $C_{max}$  inférieur à une borne supérieure sauvegardée dans l'état  $|U_b\rangle$ . Il utilise le principe de l'inversion de la phase pour marquer les états qui représentent une solution avec un signe négatif. Le schéma général de l'oracle est décrit dans 3.3.

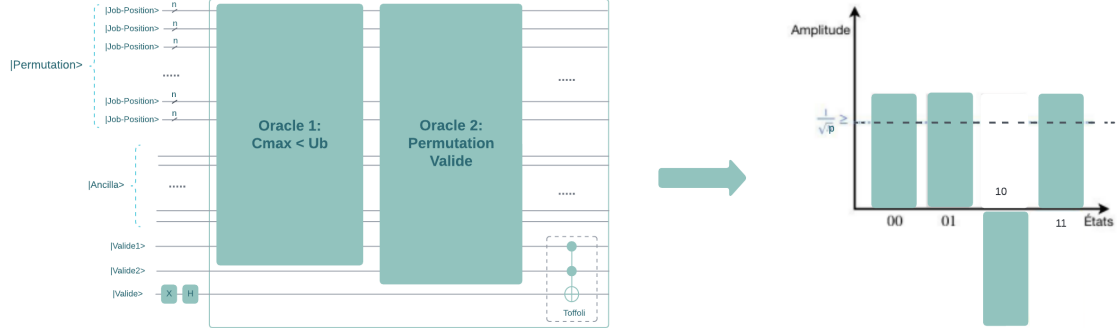


FIGURE 3.3 – Oracle

Afin de pouvoir réaliser cet oracle, nous avons divisé le problème en deux sous-problèmes et par conséquent deux oracles. Le premier oracle bascule l'état du registre  $|Valide_1\rangle$  vers  $|1\rangle$  si l'ordonnancement possède un  $C_{max}$  inférieur à la valeur de l'état  $|U_b\rangle$ .

$$|\psi_{21}\rangle = O_1|x\rangle|Valide_1\rangle = O_1|x\rangle|0\rangle = \begin{cases} |x\rangle|1\rangle & \text{Si le makespan de } x \text{ est inférieur à } U_b \\ |x\rangle|0\rangle & \text{Sinon} \end{cases}$$

Le deuxième oracle bascule l'état du registre  $|Valide_2\rangle$  vers  $|1\rangle$  si l'ordonnancement re-

présente un ordonnancement valide comme suit :

$$|\psi_{22}\rangle = O_2|x\rangle|Valide_2\rangle = O_2|x\rangle|0\rangle = \begin{cases} |x\rangle|1\rangle & \text{Si } x \text{ représente un ordonnancement valide} \\ |x\rangle|0\rangle & \text{Sinon} \end{cases}$$

Après application des deux transformations  $O_1$  et  $O_2$ , la troisième étape consiste principalement à inverser la phase des états marqués par les deux oracles. Nous avons utilisé l'opérateur Toffoli 2 tel que les deux qubits  $|Valide_1\rangle$  et  $|Valide_2\rangle$  représentent les qubits de contrôles et un troisième qubit  $|V\rangle$  initialisé dans l'état  $|-\rangle$  comme cible. La partie du circuit qui permet de réaliser l'inversion de signe des solutions est présentée dans la figure 3.4. En effet, si les deux qubits  $|Valide_1\rangle$  et  $|Valide_2\rangle$  sont dans l'état  $|1\rangle$ , l'application de l'opérateur Toffoli nous permet d'inverser la phase des états associés comme décrit dans l'équation suivante :

$$|\psi_2\rangle = (-1)^{O(s)}|x\rangle|-\rangle = (-1)^{Valide_1 \times Valide_2}|x\rangle\left|\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right\rangle$$

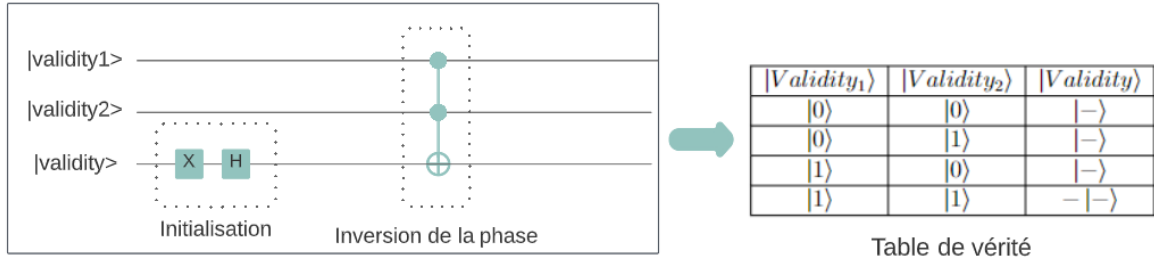


FIGURE 3.4 – Inversion de la phase

Dans ce qui suit, nous allons présenter les deux oracles en détail :

— **Oracle 1 : Est-ce que c'est une meilleure solution ?**

L'objectif principal de cet oracle est d'identifier les ordonnancements qui ont un  $|C_{max}\rangle$  inférieur à une borne  $|U_b\rangle$ . Le circuit implémenté doit, en premier lieu, calculer la valeur du  $|C_{max}\rangle$  associée à chaque permutation puis basculer l'état  $|Valide_1\rangle$  vers l'état  $|1\rangle$  si la valeur de  $|C_{max}\rangle$  est inférieur à la valeur de  $|U_b\rangle$ . Il est à noter que grâce au principe de la superposition, nous avons un parallélisme massif ce qui implique que tous les traitements sont effectués en parallèles.

1. **Le temps de fin d'exécution  $|C_{ij}\rangle$**

Nous aurions besoin de définir un opérateur  $\Omega$  qui permet de calculer le temps de fin d'exécution du job affecté à la position  $i$  sur la machine  $j$  dans un ordonnancement  $|S_k\rangle$ . l'effet de l'application de l'opérateur  $\Omega$  peut être décrit comme suit :

$$\Omega(|C_{i-1,j}\rangle|C_{i,j-1}\rangle|0\rangle_q|0\rangle_1) = |C_{i-1,j}\rangle|C_{i,j-1}\rangle|C_{i,j}\rangle|cmp\rangle$$

tel que :

- $|C_{i,j-1}\rangle$  est un registre quantique avec  $q$  qubits, qui représente le temps de fin de la tâche à la position  $i$  sur la machine  $j - 1$ .
- $|C_{i-1,j}\rangle$  est un registre quantique avec  $q$  qubits, qui représente le temps de fin de la tâche à la position  $i - 1$  sur la machine  $j$ .
- $|C_{i,j}\rangle$  représente un registre quantique avec  $q$  qubits, qui représente le temps de fin de la tâche à la position  $i$  sur la machine  $j$ .
- $|cmp\rangle$  représente un registre quantique avec un seul qubit et qui sera mis à 1 si la valeur de  $|C_{i,j-1}\rangle$  est supérieure à la valeur de  $|C_{i,j-1}\rangle$ .

Nous commençons par construire l'opérateur qui nous permet de calculer les temps de complétions  $|C_{i,j}\rangle$ , ce dernier est exprimé comme la somme du temps d'exécution  $P_{ij}$  et le temps de début d'exécution  $S_{i,j}$ , tel que  $S_{i,j}$  est égal au maximum entre les valeurs des deux états  $|C_{i-1,j}\rangle$  et  $|C_{i,j-1}\rangle$ . La première étape consiste à initialiser l'état  $|C_{i,j}\rangle$  par la valeur du temps d'exécution  $P_{i,j}$ . Pour ce faire, nous exploitons le principe d'intrication quantique en utilisant la porte CNOT 1 tel que les qubits de registres job-position seront utilisés comme des qubits de contrôle et les qubits des registres  $|C_{i,j}\rangle$  comme des qubits cibles. Si la valeur des deux qubits de contrôle est égale à  $|1\rangle$  alors l'état de qubit cibles vas être basculé.

**Exemple :** Supposons que la tâche 3 (11 en binaire) soit assignée à la position  $i$ , et que son temps de traitement  $P_{i,j}$  soit égal à 6 (0110 en binaire), nous utiliserons le qubit de job-position comme contrôle et les 3<sup>ème</sup> et 2<sup>ème</sup> qubits du temps d'achèvement comme cible. La figure 3.5 représente un exemple d'un circuit utilisé pour l'initialisation des registres  $|C_{i,j}\rangle$ .

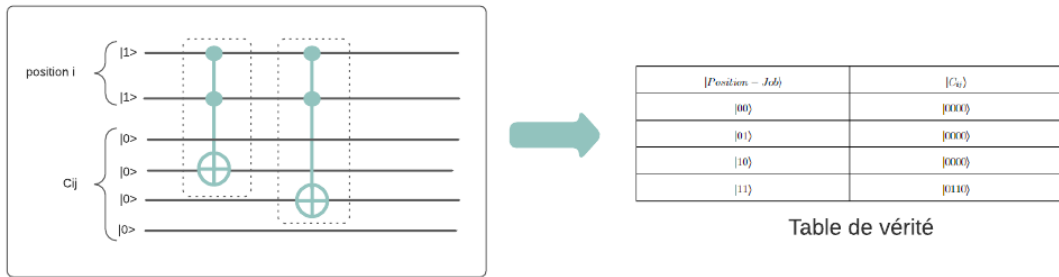


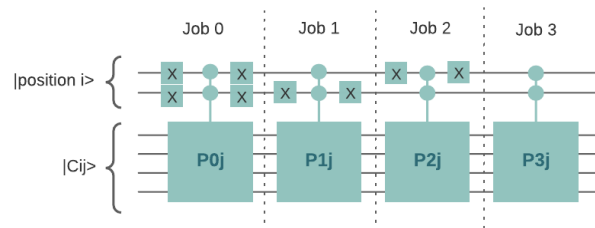
FIGURE 3.5 – L'initialisation des registres  $C_{ij}$

Les registres  $|position_i - job\rangle$  sont en superposition de tous les jobs possibles, car au début, nous ne savons pas quel job est affecté à quelle position, ce qui implique que les registres  $|C_{i,j}\rangle$  doivent être initialisés en un état de superposition de tous les temps d'exécutions possibles des jobs qui sont affectés à la position  $i$  sur la machine  $j$ .



Prenons comme exemple, une instance PFSP avec quatre jobs. Notre circuit sera conçu comme le circuit illustré dans la figure 3.6, tel que le circuit quantique prend en considération le cas où la position  $i$  est en superposition de tous les états possibles. Par conséquent l'état  $|C_{ij}\rangle$  sera préparé en superposition de tous les temps de traitement  $P_{0j}, P_{1j}, P_{2j}, P_{3j}$  attribués à la machine correspondante en fonction des tâches  $J_0, J_1, J_2, J_3$ . L'état initial des registres est décrit par l'équation suivante :

$$|position_i - job\rangle_n |C_{ij}\rangle_p = \frac{1}{2}(|job_0\rangle + |job_1\rangle + |job_2\rangle + |job_3\rangle) |0\rangle_p$$

FIGURE 3.6 – Initialisation de  $|C_{ij}\rangle$ 

Après l'application du circuit d'initialisation, nous obtenons l'état suivant :

$$|position_i job\rangle |C_{ij}\rangle = \frac{1}{2}(|job_0\rangle |P_{0j}\rangle + |job_1\rangle |P_{1j}\rangle + |job_2\rangle |P_{2j}\rangle + |job_3\rangle |P_{3j}\rangle)$$

Le pseudo algorithme qui nous permet de générer l'opérateur d'initialisation des  $|C_{ij}\rangle$  est décrit dans 3. La deuxième étape consiste à trouver le temps de début  $|S_{ij}\rangle$  et pour cela deux approches différentes peuvent être utilisées :

- (a) La première consiste à utiliser le circuit qui retourne le maximum entre  $|C_{i,j-1}\rangle$  et  $|C_{i-1,j}\rangle$  mais qui nécessite des registres quantiques supplémentaires pour sauvegarder le max et des opérateurs quantiques en plus.
- (b) La seconde approche consiste à utiliser le circuit de comparaison présenté dans la section suivante et d'utiliser le résultat de la comparaison comme qubit de contrôle pour choisir avec quel temps de complétion l'addition est appliquée.

**Algorithme 3 :** Algorithme de génération de l'opérateur  $init |C_{ij}\rangle$ 


---

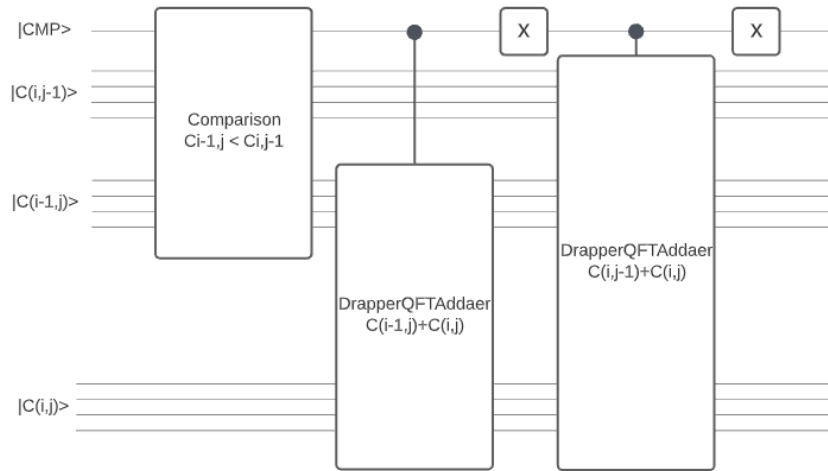
```

1 procedure Initialisation  $|C_{ij}\rangle$ 
  Input :
  q : Nombre de qubits pour représenter  $|C_{ij}\rangle$  ,
  n : Nombre de qubits pour représenter  $|position - job\rangle$  ,
  x : Liste des temps d'exécution de tous les job sur la machine j
  Output : Opérateur quantique
2 Begin
3 Création des registres quantique et du circuit quantique.
4 for chaque job do
5   xb  $\leftarrow$  Convertir le temps d'exécution de job en binaire
6   controles  $\leftarrow$  Générer la liste des qubits de controle
7   cibles  $\leftarrow$  Générer la liste des qubits Cibles
8   qc.CNOT(controle,cibles)
9   Convertir qc en un opérateur quantique
10 end
11 end procedure

```

---

Nous avons choisi de travailler avec la deuxième approche, car elle nécessite moins de qubits par rapport à la première approche, et afin d'appliquer l'addition, nous avons choisi de travailler avec l'additionneur quantique "Drapper Quantum" présenté dans **Draper (2000)**. Ce choix est justifié par le fait que ce circuit nécessite moins de nombre de qubits par rapport aux autres additionneurs. Nous aurons besoin seulement d'un seul qubit en plus pour sauvegarder la retenue et le résultat de l'addition sera retourné dans le registre  $|C_{i,j}\rangle$ .

FIGURE 3.7 –  $|C_{ij}\rangle$ 

La figure 3.7 montre le circuit permettant de calculer  $|C_{i,j}\rangle$ . La transformation de

comparaison bascule l'état  $|cmp\rangle$  vers  $|1\rangle$  si la valeur de  $|C_{i,j-1}\rangle$  est inférieure à  $|C_{i-1,j}\rangle$ . Après le qubit cmp va être utilisé comme un qubit de contrôle pour activer une des deux additions. Si  $|cmp\rangle$  est à l'état  $|1\rangle$  on active l'addition avec  $|C_{i-1,j}\rangle$  sinon on active l'autre addition. Le pseudo algorithme permettant de générer l'opérateur qui nous permet de retourner le temps de complétion est présenté ci-dessous :

---

**Algorithme 4 :** Pseudo Algorithme de génération de l'opérateur  $\Omega$

---

```

1 procedure l'opérateur  $\Omega$ 
  Input : q : Nombre de qubits pour représenter  $|C_{ij}\rangle$ 
  Output : Opérateur quantique
2 Begin
3  $c_{i,j-1} \leftarrow \text{QuantumRegister}(q)$ 
4  $c_{i,j} \leftarrow \text{QuantumRegister}(q)$ 
5  $c_{i-1,j} \leftarrow \text{QuantumRegister}(q)$ 
6  $\text{controle} \leftarrow \text{QuantumRegister}(1)$ 
7  $qc \leftarrow \text{QuantumCircuit}(n,q)$ 
8  $nbr - job \leftarrow 2^n$ 
9 Initialiser le registre  $c_{i,j}$ 
10 Ajouter opérateur de comparaison
11 Générer un opérateur d'addition contrôlé par le qubit de contrôle
12 Ajouter opérateur d'addition pour faire addition entre  $c_{i-1,j}$  et  $c_{i,j}$ 
13 Appliquer l'opérateur X sur le qubit de contrôle
14 Générer un opérateur d'addition contrôlé par le qubit de contrôle
15 Ajouter opérateur d'addition pour faire addition entre  $c_{i,j-1}$  et  $c_{i,j}$ 
16 end procedure

```

---

— **2. Calcule du makespan**

Le makespan d'un ordonnancement est égal au temps de complétion de la dernière tâche exécutée sur la dernière machine. Afin de calculer la valeur du makespan, nous devons trouver tous les  $|C_{i,j}\rangle$  de tous les jobs sous toutes les machines précédentes, ce qui implique la construction de  $((M - 1)N) + 1$  registres quantiques de q qubits pour chaque  $|C_{i,j}\rangle$ . La première machine représente un cas particulier où on se contente d'additionner le temps de traitement  $P_{i,1}$  avec le temps de complétion de la tâche précédente  $|C_{i-1,j}\rangle$ . Afin d'utiliser moins de qubits, nous n'utiliserons qu'un seul registre quantique pour calculer le temps d'achèvement  $|C_{1,j}\rangle$ . Le temps de complétion sur la première machine représente un cas particulier dont nous n'aurions pas besoin d'utiliser l'opérateur  $\Omega$  pour calculer le  $|C_{i,1}\rangle$ . Il nous suffit de faire une addition avec les  $P_{i-1,1}$ . L'algorithme permettant de générer les transformations qui nous permettent de trouver le makespan est décrit dans ce qui suit :

---

**Algorithme 5** : Algorithme de génération du circuit pour calculer le makespan

---

**Input :**q : Nombre de qubits pour représenter  $|C_{ij}\rangle$  ,

n : Nombre de qubits pour représenter une position ,

**Output :**

Opérateur quantique

```
1 Begin
2 Créer un circuit quantique
3 for position i do
4   for Machine j do
5     if  $j == 1$  then
6       | Initialiser le registre  $|T_{i1}\rangle$ 
7       | Ajouter Drapper Quantum adder
8     end
9     else
10      | Ajouter l'opérateur  $\Omega$ 
11    end
12  end
13 end
14 end procedure
```

---

- **3. Comparaison** Après le calcul du  $C_{max}$ , la prochaine étape est la comparaison de la borne supérieure avec le Cmax. Pour se faire, nous avons utilisé le même circuit de comparaison proposé dans **Vinod & Shaji (2021)**. Nous vérifions si la valeur du makespan est inférieure à la borne supérieure  $|U_b\rangle$  et nous mettons le qubit de validité à l'état  $|1\rangle$ . La comparaison entre les deux variables commence par les qubits les plus significatifs. Nous comparons les deux qubits en utilisant l'opération XOR, ce dernier est implémenté en utilisant deux paires de C-NOT de sorte que les deux qubits comparés sont utilisés comme contrôle et l'ancilla comme cible. Si les deux premiers qubits sont identiques, nous passerons à la comparaison des deux qubits suivants. Sinon, le qubit ancilla sera mis à l'état  $|1\rangle$ , ce qui signifie qu'une des variables est inférieure à l'autre. nous vérifions si la variable 1 est inférieure en utilisant la porte de Toffoli ayant le qubit de la variable 2 et le qubit d'ancilla comme des qubits de contrôle et les qubits de validité comme cible. Après chaque comparaison, l'état de l'ancilla est inversé à l'aide de la porte X afin de ne pas affecter le reste de la comparaison des qubits. La comparaison de deux variables de 3 qubits est présentée dans la figure.

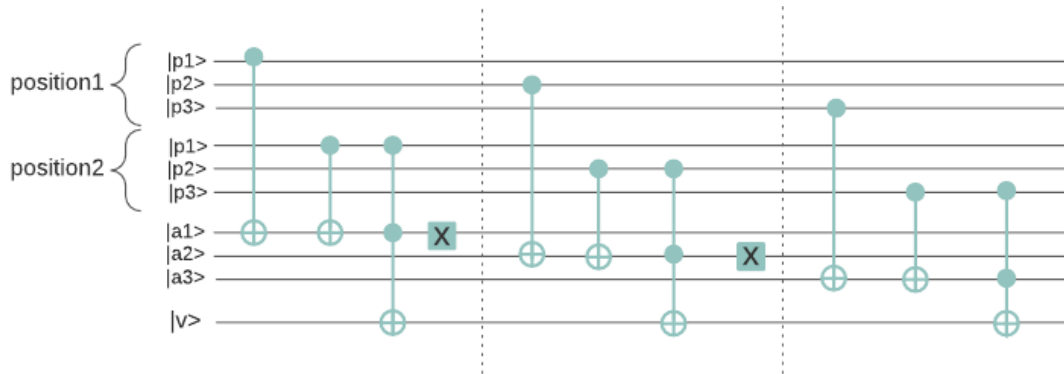


FIGURE 3.8 – Initialisation

- **Oracle 2 : vérification de la faisabilité**

Le rôle du deuxième oracle est l'identification des solutions valides, c'est-à-dire qui représentent des ordonnancements sans répétition de tâche. Afin de vérifier si une tâche a été assignée à plusieurs positions ou non, nous devons effectuer une comparaison deux à deux entre tous les registres job-position. La figure 3.8 montre le circuit utilisé pour vérifier si deux positions contiennent le même travail. Le circuit utilise l'opérateur XOR pour vérifier si deux qubits sont égaux ou non. Si les qubits sont identiques, la valeur de l'ancilla sera égale à 0, sinon elle sera égale à 1. La dernière étape consiste à effectuer un "ou" logique entre les registres ancilla et si la sortie est égale à 1, cela signifie qu'il y a au moins deux qubits qui sont différents, donc les deux positions ne comportent pas la même tâche. Le circuit présenté dans la figure 3.9 est réutilisé pour vérifier que chaque deux positions sont différents.

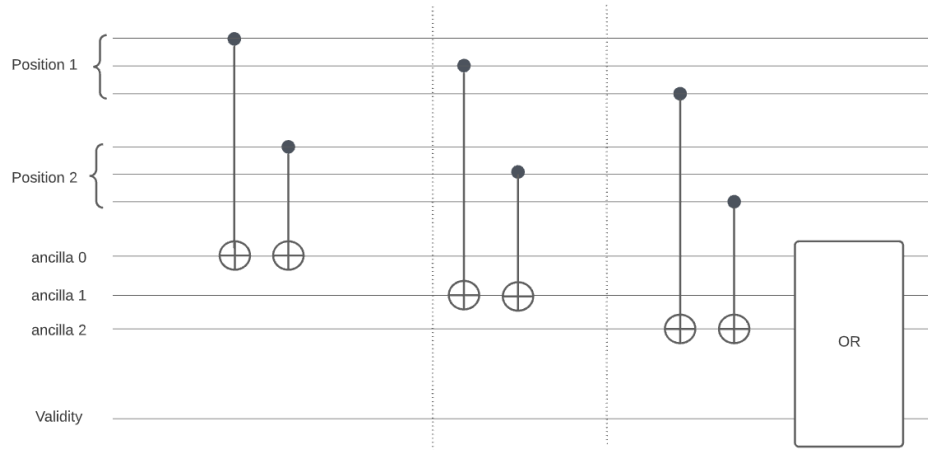


FIGURE 3.9 – Vérification si deux positions comportent le même job

### 3.5.3 Étape 3 : L'opérateur de diffusion

Présentons maintenant la dernière partie du circuit qui permet d'amplifier l'amplitude des états qui représentent des solutions et de diminuer les amplitudes des états qui restent. Cet opérateur utilise le principe de l'inversion autour de la moyenne tel que dans notre cas la moyenne est égal à  $\frac{1}{\sqrt{2^p}}$ . En effet, l'opérateur Grover entraîne une augmentation (respectivement diminution) de la probabilité d'apparition des états ayant une amplitude inférieure (respectivement supérieure) à la moyenne. Nous avons utilisé le même opérateur présenté dans **Grover (1996)**. Comme le montre la figure 3.10, l'opérateur de diffusion a augmenté la probabilité de l'état  $|01\rangle$  dont le signe était inversé par l'oracle et il a rééquilibré les valeurs des amplitudes des états restants.

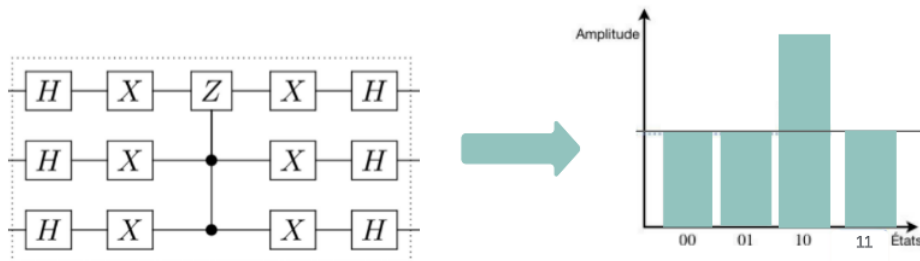


FIGURE 3.10 – L'opérateur de la diffusion

Le circuit de Grover applique les deux opérateurs oracle et diffusion plusieurs fois afin d'augmenter suffisamment les probabilités d'apparition des solutions, dans les meilleurs cas. Nous pouvons obtenir une superposition des solutions seulement si le nombre d'itérations de Grover (ou nombre de rotations) était bien défini. Trouver le nombre d'itérations suffisant est difficile si le nombre de solutions n'est pas connu auparavant.

### 3.6 Phase 3 : Optimisation globale avec l'algorithme de Grover

Dans cette partie, nous allons présenter deux méthodes différentes qui nous permettent d'utiliser le circuit de Grover pour l'optimisation global. L'idée de base est de décomposer le problème en deux parties, la première partie consiste en l'estimation de la borne supérieure et le nombre de rotations de l'opérateur de Grover alors que la deuxième partie est la recherche des solutions ayant un  $C_{max}$  inférieur à la borne supérieure définie par la première partie comme le montre la figure 3.11. La première approche consiste à utiliser l'approche GAS **Gilliam et al. (2021)** pour trouver une solution optimale alors que la deuxième approche utilise la recherche binaire pour initialiser la borne supérieure.

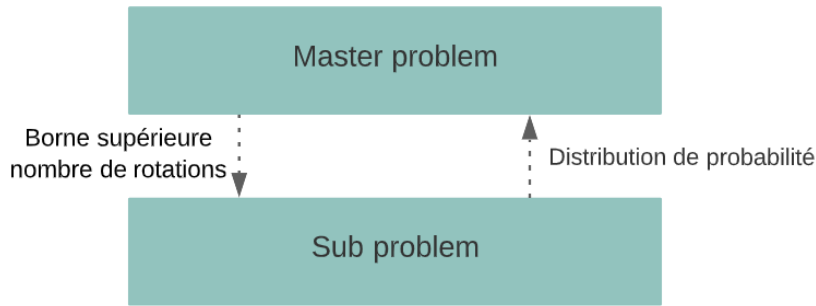


FIGURE 3.11 – Décomposition du problème de Flow Shop

#### 3.6.1 Algorithme basé sur l'approche GAS

L'algorithme de Grover nous permet d'avoir une solution à un problème d'optimisation où la borne supérieure est fixée. Dans cette partie, notre intention est d'utiliser la recherche de Grover de manière répétée dans une méthode d'optimisation globale de type recherche adaptative. Les méthodes de recherche adaptative produisent, où tentent de produire, une séquence améliorée d'échantillons. L'algorithme utilise comme paramètre une séquence  $(r_n : n = 1, 2, \dots)$  de nombres de rotations qui représente le nombre d'itérations de l'opérateur de Grover.

Le processus commence par une solution initiale qui est dans notre cas choisi aléatoirement ou alors générées par des heuristiques comme NEH. Ensuite, la tâche de l'algorithme de Grover consiste à trouver toute solution valide dont le coût est meilleur que la solution initiale. Si le résultat de la recherche est positif, nous remplaçons la solution initiale par la nouvelle solution et nous continuons l'itération. À chaque itération, l'algorithme de Grover utilise la meilleure solution déjà vue comme seuil et le paramètre  $r_n$  comme nombre d'itérations de l'opérateur de Grover (oracle+diffusion). Le critère d'arrêt de l'algorithme est le parcours de tous l'espace de recherche ou le dépassement d'un nombre d'itérations définit auparavant. Les étapes de l'algorithme sont décrit dans l'organigramme qui décrit

le processus de GAS 3.12.

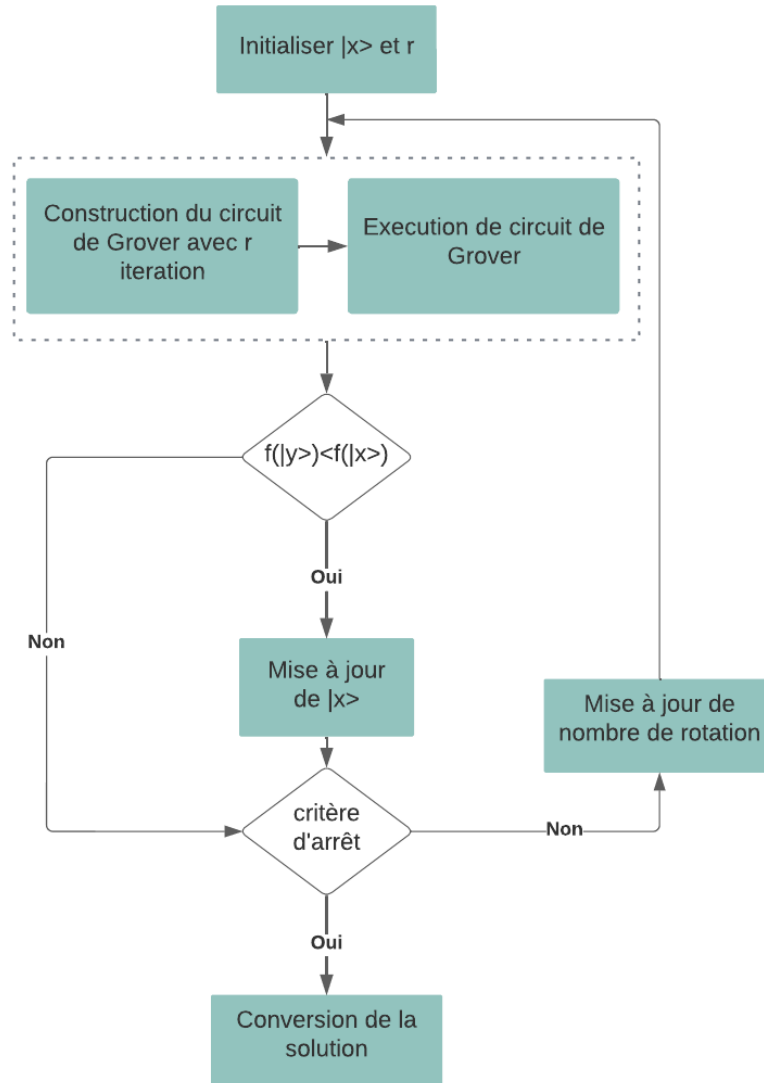


FIGURE 3.12 – Organigramme de la méthode GAS

Le choix du nombre de rotations de Grover à chaque itération est une étape très importante. En effet, si ce paramètre est bien choisi, la probabilité d'apparition d'une solution améliorée est grande, mais même dans ce cas le succès n'est pas garanti, car la probabilité des autres états restent différentes du zéro. Si le nombre de solutions de flow shop est connu auparavant, nous pouvons déterminer la meilleure valeur de  $r_n$  mais malheureusement nous ne pouvons pas le déterminer sans la recherche de toutes les solutions. Par conséquent, nous avons opté pour la méthode de **Baritomba *et al.* (2005)** qui propose une stratégie pour la sélection du nombre d'itérations basée sur une sélection aléatoire d'un nombre parmi un ensemble de nombres entiers. Cet ensemble commence avec  $\{0\}$  comme seul élément. Lorsque la recherche ne réussit pas à trouver une meilleure solution, l'algorithme ajoute d'autres éléments à un maximum de  $\{0, \dots, [m - 1]\}$  à chaque étape



de recherche, jusqu'à ce qu'une meilleure solution que la précédente soit trouvée. De cette façon, l'ensemble comprend plus de nombres entiers comme éléments. Ainsi, la probabilité de sélectionner le bon nombre d'itérations pour que la recherche réussisse augmente. La valeur de  $m$  est mise à jour à chaque étape par  $\min(\lambda * m, \sqrt{2^{N*n}})$ , où  $\lambda$  est donné comme un paramètre et  $2^{N*n}$  est le nombre total d'éléments de l'espace de recherche. Par conséquent,  $m$  ne doit pas excéder  $\sqrt{2^{N*n}}$ , qui est le nombre optimal de rotations pour trouver un élément spécifique dans un ensemble de  $2^{N*n}$  éléments lorsque le nombre de solutions soit égale à un.

L'algorithme de l'approche GAS est décrit dans ce qui suit :

---

**Algorithme 6** : Algorithme de recherche adaptative basée sur Grover

---

```
1 procedure GAS
2 Initialiser  $x$  aléatoirement
3  $x_1 \leftarrow x$ 
4  $y_1 \leftarrow f(x)$ 
5  $m \leftarrow 1$ 
6 Choisir une valeur de  $\lambda$  (la valeur proposée dans les articles est 8/7)
7 for le critère d'arrêt n'est pas atteint do
8   Choisir d'une façon aléatoire le nombre de rotation  $r_i$  de l'ensemble
       $\{0, \dots, m - 1\}$ 
9   Construire le circuit de grover avec  $r_i$  comme nombre de rotation et  $y_i$  comme
      une borne superieur
10  Exécuter l'algorithme de Grover et retourner
11  Mise à jour de  $x$  et  $y$ 
12  if  $y < y_i$  et valide( $x$ ) then
13     $x_i \leftarrow x$ 
14     $y_i \leftarrow y$ 
15     $m \leftarrow 1$ 
16  end
17  else
18     $m \leftarrow \lambda * m$ 
19  end
20   $i = i + 1$ 
21 end
22 end procedure
```

---

Nous avons changé la partie d'initialisation telle qu'au lieu d'initialiser la borne au début de manière aléatoire, nous utilisons une heuristique (NEH) pour l'initialiser.

### 3.6.2 Algorithme basé sur la Recherche binaire

Cette méthode est basée sur la recherche binaire classique, elle a été utilisée pour la résolution du problème de sac à dos **Baykov & Protasov (n.d.)**. Contrairement à la méthode GAS, cette approche utilise la recherche binaire pour initialiser la borne supérieure à chaque itération. Notre borne supérieure initiale est initialisée par la somme de tous les temps d'exécution et la borne inférieure par la somme de tous les temps d'exécution sur la première machine plus le minimum entre les temps d'exécution sur le reste des machines. Concernant la partie de génération du nombre de rotations, l'algorithme génère une suite de rotations allant jusqu'à  $\sqrt{2^{N*n}}$  comme décrit dans l'algorithme suivant :

---

**Algorithme 7** : Algorithme de recherche binaire basée sur Grover

---

```
1 procedure BSG
2 Initialiser bs
3 Initialiser bi
4 while  $bi < bs$  do
5      $m = \frac{bs+bi}{2}$ 
6     while  $i < \sqrt{2^{N*n}}$  do
7         Construire le circuit de Grover avec  $i$  comme nombre de rotation et  $m$ 
           comme une borne supérieure
8         Exécuter l'algorithme de Grover et retourner
9         Mise à jour de x et y
10        if  $y < y_i$  et valide( $x$ ) then
11             $x_i \leftarrow x$ 
12             $y_i \leftarrow y$ 
13             $i \leftarrow 1$ 
14        end
15        else
16             $i \leftarrow 2 * i$ 
17        end
18    end
19     $bs = m$ 
20 end
21 end procedure
```

---

## 3.7 Conclusion

Dans ce chapitre, nous avons étudié le problème d'atelier de type FlowShop de permutation (PFSP). Nous avons proposé l'approche GroverFSP basé sur l'algorithme de Grover. L'objectif de Grover était de trouver des solutions satisfaisant deux contraintes : (1) un ordonnancement valide ; (2) possède un Cmax inférieur à  $U_{bs}$ . Ce circuit est exploré pour faire l'optimisation globale en utilisant deux versions, une version basée sur la recherche binaire (BSG) et l'autre version est basée sur l'algorithme GAS.

Dans la chapitre suivant, nous allons présenter un autre type de méthodes appelé les méthodes variationnelles.

# Chapitre 4

## Approches quantiques basée sur les algorithmes variationnels

### 4.1 Introduction

L’algorithme présenté dans le chapitre précédent nécessite un nombre très important de qubits et de portes quantiques ainsi que des codes de correction d’erreur, mais les ordinateurs quantiques disponibles aujourd’hui ne sont pas encore au point pour pouvoir exécuter un tel algorithme.

Dans cette partie, nous allons présenter un autre type de méthode appelé les méthodes hybrides (variationnelles) qui sont adaptées pour être exécuter sur les ordinateurs de l’ère NISQ. Le second algorithme proposé, nommé VQA-FSP, passe par trois phases principales. La première phase consiste en la modélisation mathématique du PFSP sous forme quadratique binaire sans contraintes (QUBO). Dans ce contexte, nous avons proposé deux modèles mathématiques : (1) un modèle approximatif basé sur la modélisation indice-position, (2) une approximation du flowshop vers le problème du voyageur de commerce (TSP). Ensuite, la deuxième phase consiste à mapper notre problème vers un Hamiltonien du modèle d’Ising. Pour se faire, nous avons utilisé un convertisseur permettant d’obtenir l’Hamiltonien depuis la forme QUBO. La dernière étape consiste en l’utilisation des méthodes variationnelles pour la résolution du PFSP. Nous avons commencé par choisir les paramètres de ces méthodes (optimiseur classique, ansatz...). Ensuite, nous avons implémenté trois approches différentes : VQA, QAOA et une version modifiée du QAOA appelée QAOA-Swap.

La figure 4.1 montre l’architecture générale de l’approche.

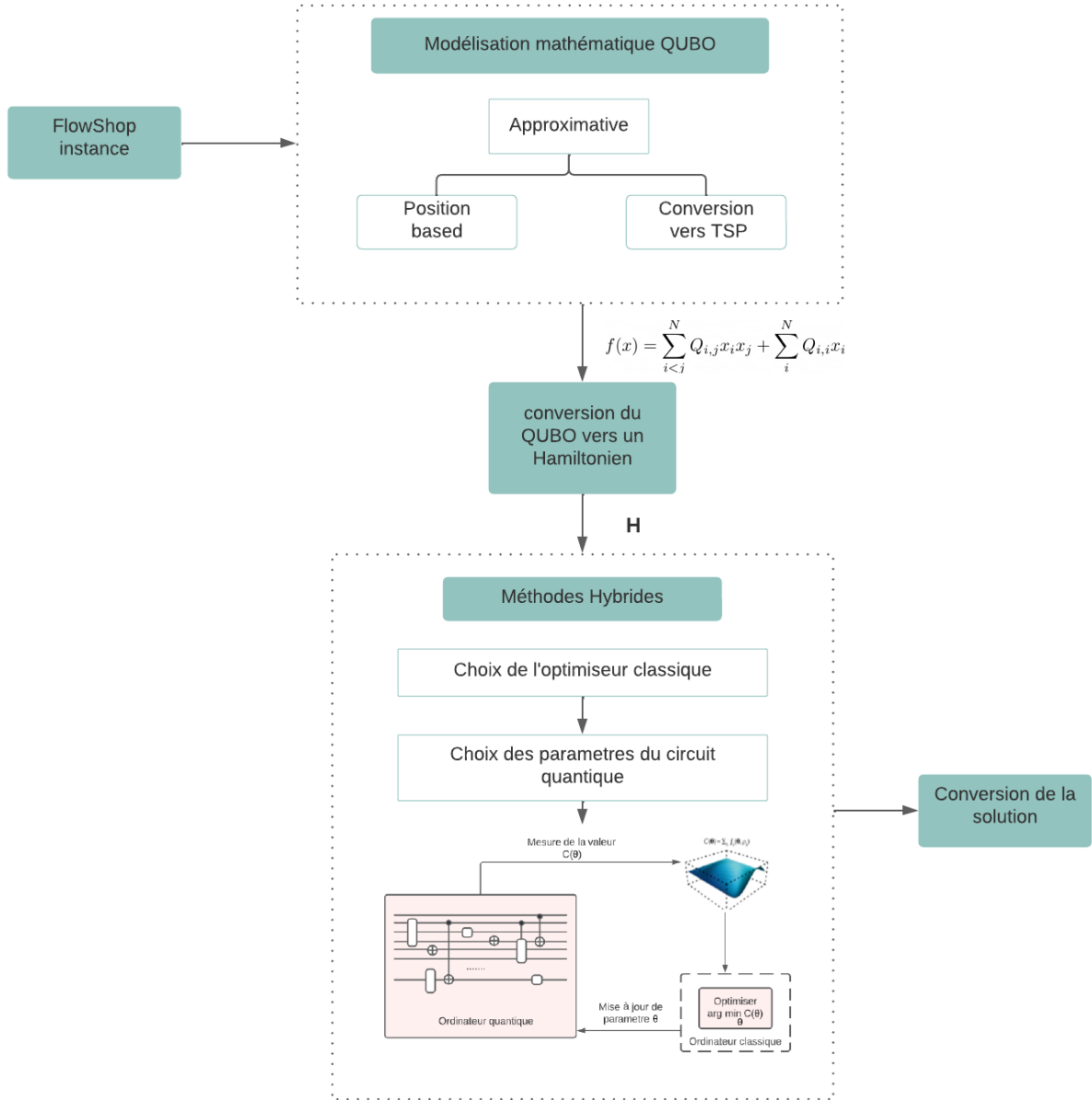


FIGURE 4.1 – Architecture générale de l'approche

Dans ce qui suit, nous allons présenter en détail chaque phase de l'approche proposée.

## 4.2 Modélisation mathématique

Dans cette section, nous allons modéliser sous forme quadratique binaire sans contraintes (QUBO) 2.3 le problème du PFSP sur  $m$  machines  $F_m |perm| C_{max}$ . Le premier modèle est basé sur l'idée d'approximer le PFSP vers un problème du voyageur de commerce (TSP) et ensuite nous utiliserons la modélisation QUBO du TSP. Le second modèle *QUBO* représente une modélisation approximative basée sur le modèle indice-position. Il existe dans la littérature des modélisations exactes proposées pour le problème de Job-Shop qui peuvent être adaptées facilement à notre problème, mais le nombre de qubits nécessaire

pour la résolution de ces derniers est très grand **Shimada et al. (2021); Venturelli et al. (2016)**.

Considérant un PFSP avec un ensemble de job  $J$  de  $N$  jobs tel que  $J = \{j_1, j_2, \dots, j_N\}$  qui doivent être exécutés sur un ensemble  $M$  de  $m$  de machines  $M = \{M_1, M_2, \dots, M_m\}$ . Chaque tâche consiste en une séquence d'opérations qui doivent être exécutées dans un ordre prédéfini sur toutes les machines  $j_n = \{O_1, O_2, \dots, O_m\}$ .

### 4.2.1 Un modèle QUBO basé sur le modèle QUBO du TSP

Le premier modèle est basé sur l'idée d'approximer le problème du PFSP par un problème de voyageur de commerce (TSP) qui a pour objectif de déterminer le plus court chemin pour visiter une liste de villes avec la contrainte de ne passer qu'une seule fois par chacune d'elle. Le modèle QUBO du TSP est facile à déterminer et nous pouvons l'exprimer comme suit :

**La variable de décision**

$$x_{ip} = \begin{cases} 1 & \text{si le voyageur passe par la ville } i \text{ en } p\text{-ieme position ,} \\ 0 & \text{sinon} \end{cases}$$

**La fonction objectif :**

**Minimiser**  $E$

$$E = E_{contraintes} + E_{cout} \quad (4.1)$$

La partie  $E_{cout}$  représente la distance totale qu'on cherche à minimiser, elle est égale à :

$$E_{cout} = \sum_{i,j}^N d_{ij} \sum_p^{N-1} x_{i,p} x_{j,p+1} \quad (4.2)$$

tel que  $d_{i,j}$  représente la distance entre la ville  $i$  et la ville  $j$ .

La partie  $E_{contraintes}$  est la partie qui nous assure que les solutions trouvées sont des solutions valides, c'est-à-dire, qui respectent la contrainte de passer une et une seule fois par une ville.

$$E_{contraintes} = P \sum_p^N (1 - \sum_i^N x_{ip})^2 \quad (4.3)$$

$$+ P \sum_i^N (1 - \sum_p^N x_{ip})^2 \quad (4.4)$$

La première partie 4.3 de la solution nous permet de pénaliser les variables dont deux villes sont affectées à la même position et l'expression 4.4 pour pénaliser les variables dont une ville est assignée à deux positions différentes. La variable  $P$  représente une constante pour assurer que les variables qui ne représentent pas de solutions valides et qui possèdent une

valeur d'évaluation assez grande par rapport aux solutions valides. Par conséquent, nous avons choisi d'initialiser  $P$  par une valeur qui maximise la fonction coût 4.2. Nous pouvons voir que si toutes les distances sont positives, la solution qui maximise la fonction de coût  $E_{cout}$  est la solution dont toutes les variables de décisions sont égales à 1 et elle peut être écrite comme suit :

$$P = \sum_{i \neq j}^N |d_{i,j}| \sum_p^{N-1} 1$$

Mais même en utilisant cette pénalité, nous pouvons avoir des solutions non faisables. En s'inspirant de la logique du docplex pour la génération des pénalités, nous avons initialisé notre pénalité par :

$$P = a \sum_{i \neq j}^N |d_{i,j}| \sum_p^{N-1} 1 + cste$$

tel que  $a$  et  $cste$  représentent deux constantes supérieures à 1 choisit aléatoirement.

Le modèle QUBO de FSP est décrit comme suit :

$$E = \sum_{i \neq j}^N d_{ij} \sum_p^{N-1} x_{i,p} x_{j,p+1} + P \sum_p^N (1 - \sum_i^N x_{ip})^2 + P \sum_i^N (1 - \sum_p^N x_{ip})^2 \quad (4.5)$$

### Mapper PFSP vers TSP

Afin de convertir le PFSP vers le TSP, nous allons considérer les jobs comme des villes puis calculer les distances  $d_{u,v}$  entre chaque deux jobs  $u$  et  $v$  en utilisant les approximations présentées dans le tableau ci-dessous 4.1.

Approximation	Référence	Distance
1	Widmer et Hertz	$d_{u,v} = p_{i1} + \sum_{i=2}^m (m-i)  p_{u,v} - p_{i-1,v}  + p_{m,v}$
2	Gupta	$d_{u,v} = CT_m(u, v) - \sum_{i=1}^m p_{u,i}$ $CT_j(u, v) = \max\{CT_j - 1(u, v), \sum p_{i,v}\}$
3	Stinson et Smith	$d_{u,v} = \sum_{i=0}^m \max\{(p_{i,u} - p_{i-1,v}), 0\} + 2 * \min\{(p_{i,u} - p_{i-1,v}), 0\}$
4	Mocellin	$d_{u,v} = UB X(m)_{u,v}$ $UB X(1)_{u,v} = 0$ $UB X(K+1)_{u,v} = \max\{0, UB X(K+1)_{u,v} + (p_{k,v} - p_{k+1,u})\}$
5	Stinson et Smith	$d_{u,v} = \sum_{i=2}^m  p_{i,u} - p_{i-1,v} $

TABLE 4.1 – Approximation de FSP vers TSP

**Exemple :** Considérons une instance du PFSP avec trois jobs et trois machines avec les temps d'exécution présentés dans le tableau suivant :

	M1	M2	M3
J1	1	2	5
J2	2	3	1
J3	4	7	6

Afin de convertir le PFSP vers un TSP, nous allons considérer chaque job comme une ville :  $J_1 \rightarrow V_1, J_2 \rightarrow V_2$  et  $J_3 \rightarrow V_3$ , et calculer les distances entre ces villes en utilisant l'approche 5 présentée dans le tableau 4.1. Notre nouvelle instance TSP est décrite dans le tableau suivant :

	V1	V2	V3
V1	-	1	4
V2	3	-	7
V3	10	8	-

Le modèle QUBO correspondant est exprimé comme suit :

**Min**  $E$  tq

$$E = \sum_{i,j}^3 d_{ij} \sum_p^2 x_{i,p} x_{j,p+1} + P \sum_p^3 (1 - \sum_i^3 x_{ip})^2 + P \sum_i^3 (1 - \sum_p^2 x_{ip})^2 \quad (4.6)$$

$$\begin{aligned} E = & d_{1,2} * (x_{1,1}x_{2,2} + x_{1,2}x_{2,3}) + d_{1,3} * (x_{1,1}x_{3,2} + x_{1,2}x_{3,3}) + d_{2,1} * (x_{2,1}x_{1,2} + x_{1,2}x_{2,3}) + \\ & d_{2,3} * (x_{2,1}x_{3,2} + x_{2,2}x_{3,3}) + d_{3,1} * (x_{3,1}x_{1,2} + x_{3,2}x_{2,3}) + d_{3,2} * (x_{3,1}x_{2,2} + x_{3,2}x_{2,3}) \\ & + P * (1 - x_{1,1} - x_{2,1} - x_{3,1})^2 + P * (1 - x_{1,2} - x_{2,2} - x_{3,2})^2 + P * (1 - x_{1,3} - x_{2,3} - x_{3,3})^2 \\ & + P * (1 - x_{1,1} - x_{1,2} - x_{1,3})^2 + P * (1 - x_{2,1} - x_{2,2} - x_{2,3})^2 + P * (1 - x_{3,1} - x_{3,2} - x_{3,3})^2 \end{aligned}$$

La pénalité  $P$  est initialisée par :

$$P = 2 \left( \sum_{i \neq j}^3 d_{i,j} \sum_p^2 1 \right) + 2 = 138$$

Il est à noter que nous considérons  $x_{i,j} = x_{i,j}^2$  car  $x_{i,j} \in 0, 1$ . Après développement de la fonction objectif  $E$ , nous obtenons la fonction suivante :

$$\begin{aligned} E = & -138 * x_{1,1}^2 + 138 * x_{1,1}x_{1,2} + 138 * x_{1,1}x_{1,3} + 138 * x_{1,1}x_{2,1} + 1 * x_{1,1}x_{2,2} + 138 * x_{1,1}x_{3,1} + \\ & 4 * x_{1,1}x_{3,2} - 138 * x_{1,2}^2 + 138 * x_{1,2}x_{2,3} + 3 * x_{1,2}x_{2,1} + 138 * x_{1,2}x_{2,2} + 2 * x_{1,2}x_{2,1} + 10 * x_{1,2}x_{3,1} + \end{aligned}$$



$$\begin{aligned}
 & 4 * x_{1,2} x_{3,3} - 138 * x_{1,3}^2 + 3 * x_{1,3} * x_{2,2} + 138 * x_{1,3} * x_{2,3} + 10 * x_{1,3} * x_{3,2} + 138 * x_{1,3} * x_{3,3} - 138 * \\
 & x_{2,1}^2 + 138 * x_{2,1} * x_{2,2} + 138 * x_{2,1} * x_{2,3} + 138 * x_{2,1} * x_{2,1} + 7 * x_{2,1} * x_{3,2} - 138 * x_{2,2}^2 + 138 * x_{2,2} * \\
 & x_{2,3} + 8 * x_{2,2} * x_{3,1} + 138 * x_{2,2} * x_{3,2} + 7 * x_{2,2} * x_{3,3} - 138 * x_{2,3}^2 + 8 * x_{2,3} * x_{3,2} + 138 * x_{2,3} * x_{3,3} - \\
 & 138 * x_{3,1}^2 + 138 * x_{3,1} * x_{3,3} - 138 * x_{3,2}^2 + 138 * x_{3,2} * x_{3,3} + 138 * x_{3,2} * x_{3,3} - 138 * x_{3,3}^2 + 414
 \end{aligned}$$

Cette expression peut être exprimée comme suit :

$$E = x^T Q x + 414 \quad (4.7)$$

tel que  $x$  représente notre vecteur de décision et  $Q$  une matrice carrée.

$$x = \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_{2,1} \\ x_{2,2} \\ x_{2,3} \\ x_{3,1} \\ x_{3,2} \\ x_{3,3} \end{bmatrix} ; Q = \begin{bmatrix} -138 & 138 & 138 & 138 & 1 & 0 & 138 & 4 & 0 \\ 0 & -138 & 138 & 3 & 138 & 2 & 10 & 138 & 4 \\ 0 & 0 & -138 & 0 & 3 & 138 & 0 & 10 & 138 \\ 0 & 0 & 0 & -138 & 138 & 138 & 138 & 7 & 0 \\ 0 & 0 & 0 & 0 & -138 & 138 & 8 & 138 & 7 \\ 0 & 0 & 0 & 0 & 0 & -138 & 0 & 8 & 138 \\ 0 & 0 & 0 & 0 & 0 & 0 & -138 & 138 & 138 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -138 & 138 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -138 \end{bmatrix}$$

Pour la séquence  $job_0 \rightarrow job_1 \rightarrow job_2$  le vecteur de décision est écrit comme suit :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

$$E(x) = x^T Q x + 414 = 8$$

### 4.2.2 Un modèle QUBO basé sur le modèle Position index

Ce modèle mathématique est inspiré du modèle mathématique linéaire du problème du flowshop présenté dans Šeda (2007). Nous allons considérer trois ensembles  $J, M$  et  $O$  tel que :

- $J$  est l'ensemble des tâches  $\{1, \dots, n\}$ .
- $M$  est l'ensemble des machines  $\{1, \dots, m\}$ .
- $O$  est l'ensemble des opérations  $\{1, \dots, m\}$ .

Dénotons les variables suivantes :

1.  $j_i$  représente le  $i$ -ième travail dans la séquence de travaux.
2.  $p_{ik}$  représente le temps de traitement du job  $j_i \in J$  sur la machine  $k$ .
3.  $v_{ik}$  représente le temps d'attente ( idle time) sur la machine  $k$  avant le début de la tâche  $j_i$ .
4.  $w_{ik}$  représente le temps d'attente (temps mort) de la tâche  $j_i$  après avoir terminer son traitement sur la machine  $k$ , en attendant que la machine  $k+1$  se libère

Notre variable de décision est ;

$$x_{ij} = \begin{cases} 1 & \text{si le job } j \text{ est affecté à la position } i \text{ dans la séquence,} \\ 0 & \text{sinon} \end{cases}$$

**Les contraintes linéaires :**

$$\forall i \in J \quad \sum_{j=1}^n x_{ij} = 1 \quad (4.8)$$

$$\forall j \in J \quad \sum_{i=1}^n x_{ij} = 1 \quad (4.9)$$

$$\forall k \in M - \{1\} \quad v_{1k} = \sum_{r=1}^{k-1} \sum_{i=1}^n p_{ir} x_{1,i} \quad (4.10)$$

$$\forall k \in M = \{m\} \quad w_{1k} = 0 \quad (4.11)$$

$$v_{i+1,k+1} = v_{i+1,k} + \sum_{j=1}^n p_{jk} x_{i+1,j} - \sum_{j=1}^n p_{j,k+1} x_{i,j} + w_{i+1,k} - w_{i,k} \quad (4.12)$$

la contrainte (6.1) exprime le fait que chaque position doit contenir un job, la contrainte (6.2) indique que chaque job est affecté à une position, la contrainte (6.3) implique que le temps d'attente du premier job sur la machine  $k$  est égale à la somme de tous les temps d'exécution des jobs sur les machines précédentes, la contrainte (6.4) exprime le fait que le premier job n'a pas besoin d'attendre pour s'exécuter sur la première machine et la

dernière contrainte nous assure la validité des ordonnancements.

**La fonction objectif :**

$$C_{max} = \sum_{i=1}^n (v_{i,m} + \sum_{j=1}^n p_{jm} x_{ij})$$

D'après la fonction objectif, on peut déduire que minimiser le  $C_{max}$  revient à minimiser la somme des  $v_{i,m}$ . En utilisant la contrainte (6.5) nous pouvons simplifier la somme des  $v_{i,m}$  par l'expression suivante :

$$\sum_{i=1}^n v_{i,m} = \sum_{i=1}^n v_{i,1} + \sum_{k=0}^{m-1} w_{N,k} + \sum_{k=1}^m \sum_{i=1}^n \left( \sum_{j=0}^n p_{j,k} x_{i+1,j} - \sum_{j=0}^n p_{j,k+1} x_{i,j} \right)$$

Nous remarquons que la dernière partie représente une fonction quadratique. Pour obtenir notre modèle QUBO, nous avons approximé notre modèle par l'expression suivante :

$$E_{cout} = \sum_{k=1}^m \sum_{i=1}^n \left( \sum_{j=0}^n p_{j,k} x_{i+1,j} - \sum_{j=0}^n p_{j,k+1} x_{i,j} \right)$$

Il reste maintenant à exprimer les deux contraintes (6.1) et (6.2) comme des pénalités et de les ajouter à notre fonction de coût pour assurer la faisabilité des solutions obtenues. Pour se faire, nous allons utiliser les règles présentées dans la section 4.3. Selon ces règles la contrainte (6.1) peut être exprimée comme suit :

$$\sum_{j=1}^n \left( \sum_{i=1}^n (x_{i,j} - 1)^2 \right) \quad (4.13)$$

La contrainte (6.2) est exprimée par l'expression suivante :

$$\sum_{j=1}^n (x_{i,j} - 1)^2 \quad (4.14)$$

Donc la partie  $E_{contraintes}$  qui permet d'assurer que la faisabilité des solutions est décrite comme suit :

$$E_{contraintes} = \sum_{j=1}^n \left( \sum_{i=1}^n (x_{i,j} - 1)^2 \right) + \sum_{i=1}^n \left( \sum_{j=1}^n (x_{i,j} - 1)^2 \right) \quad (4.15)$$

Au finale notre modèle quadratique sans contraintes est décrit par l'expression suivante :

$$E = P * E_{contraintes} + E_{cout}$$

P représente une pénalité initialisée par la somme de tous les temps d'exécution multipliée par une constante afin d'assurer que les solutions qui ne respectent pas les contraintes

soient pénalisées.

$$P = A * \sum_{i=0}^N \sum_{j=0}^M P_{i,j} \quad (4.16)$$

tel que A représente une constante.

### 4.3 L’hamiltonien

Le pont entre le PFSP et les approches quantiques variationnelles est l’Hamiltonien qui représente une matrice qui décrit l’énergie d’un système quantique. En effet, les algorithmes variationnels quantiques cherchent à trouver une approximation de l’état d’un système quantique ayant une énergie minimale.

Afin de résoudre notre problème, nous allons le voir comme un système quantique dont l’état quantique représente notre vecteur de décision tel que chaque variable de décision représente un qubit et le Hamiltonien représente notre matrice de coût. Afin d’obtenir cette matrice, nous avons utilisé un convertisseur permettant de convertir notre modèle QUBO

$$x^T Q x \quad (4.17)$$

vers un Hamiltonien du modèle Ising

$$\langle \psi(\theta) | H | \psi(\theta) \rangle \quad (4.18)$$

la mise en correspondance d’un problème d’optimisation combinatoire avec un Hamiltonien du modèle d’Ising peut être très difficile, compliqué et nécessite des connaissances spécialisées comme la vectorisation de la matrice pour exprimer la multiplication des matrices comme une transformation linéaire des matrices. Afin de faciliter cette tâche, nous avons utilisé un convertisseur de docplex pour obtenir notre Hamiltonien.

### 4.4 Algorithme variationnel quantique

Les algorithmes quantiques variationnels (VQA) dominent le calcul quantique basé sur les portes logiques, puisqu’ils sont apparus comme la stratégie principale pour obtenir un avantage quantique sur les dispositifs NISQ. L’un des principaux avantages des VQA est leur polyvalence vu qu’ils fournissent un cadre général qui peut être utilisé pour résoudre un large éventail de problèmes. Cependant, les VQA partagent certains éléments communs. Il s’agit de la fonction objectif, un circuit quantique paramétré (PQC), c’est-à-dire le circuit quantique dont les paramètres sont manipulés dans le processus de minimisation, un schéma de mesure, qui extrait les valeurs d’espérance pour évaluer la fonction objectif. Le dernier ingrédient est l’optimiseur classique qui est nécessaire pour trouver les bons

paramètres du circuit.

1. **La fonction objectif.** Un des aspects cruciaux d'un VQA est l'encodage du problème dans une fonction de perte, qui fait correspondre les valeurs des paramètres entraînables en un nombre réel :

$$f : \theta \rightarrow \mathbb{R}$$

La fonction objectif dépend du problème étudié. Minimiser cette fonction revient à trouver les meilleurs paramètres du circuit permettant de générer une solution qui minimise la fonction objectif.

2. **Le circuit paramétré** est autre élément important d'un VQA qui prépare l'état qui répond le mieux à l'objectif. Nous nous y référerons également comme l'ansatz du problème. Il existe deux approches pour définir la structure d'un ansatz : l'approche inspirée par le problème et l'approche indépendante du problème. La première, la plus courante, est employée lorsque le circuit paramétré est conçu pour s'adapter au problème spécifique. Par contre, la seconde est générique et peut être utilisée quand aucune information pertinente n'est disponible. Formellement, l'ansatz est appliqué à un état initial  $|\psi_0\rangle$ . Une fois que nous avons défini l'opération unitaire effectuée par l'ensemble du circuit comme  $U(\theta)$ , l'état final est exprimé comme suit :

$$|\psi(\theta)\rangle = U(\theta) |\psi_0\rangle$$

Généralement l'état  $|\psi_0\rangle$  est initialisé dans l'état  $|0\rangle^n$  et les paramètres du circuit sont choisis aléatoirement. Un bon ansatz est un ansatz qui peut être entraîné et qui génère tous les états possibles de l'espace de recherche.

3. **Le schéma de mesure.** afin d'estimer la valeur de la fonction objectif, nous aurons besoin de mesurer les valeurs des qubits  $|0\rangle$  ou  $|1\rangle$ .
4. **L'optimiseur classique** est le dernier élément permettant de trouver les meilleurs paramètres du circuit pour générer un état qui minimise la fonction objectif. Dans notre cas, nous avons choisi de travailler avec la méthode d'approximation stochastique des perturbations simultanées **Spall (1998)**.

Dans le cadre de la résolution du PFSP, nous avons proposé trois méthodes variationnelle quantique : (1) la méthode variational quantum Eigensolver (VQE), (2) Quantum approximate optimization algorithm (QAOA) et la dernière méthode représente une amélioration de QAOA ou nous avons introduit un changement dans l'algorithme. Dans ce qui suit, nous allons présenter ces méthodes.

### 4.4.1 Variational Quantum Eigensolver

La première méthode connu par VQE représente un algorithme variationnel très général utilisé pour la résolution des problèmes de la chimie. Nous avons utilisé cet algorithme pour trouver une approximation de la solution optimale à notre problème.

Notre problème est décrit par un Hamiltonien  $H$  comme décrit dans la partie précédente dont l'état d'énergie n'est pas connu et le circuit paramétré est décrit par la transformation linéaire  $U(\theta)$ . la fonction objectif est décrite comme suit :

$$f(\theta) = \langle \psi(\theta) | H | \psi(\theta) \rangle \quad (4.19)$$

tel que  $|\psi(\theta)\rangle$  représente l'état obtenu après l'application du circuit paramétré avec  $\theta = (\theta_1, \theta_2, \dots, \theta_q)$  sur l'état initial  $|\psi_0\rangle$ , comme exprimé dans l'équation suivante :

$$|\psi(\theta)\rangle = U(\theta) * |\psi_0\rangle \quad (4.20)$$

En minimisant  $f$ , nous allons obtenir L'ensemble des paramètres  $\theta_{opt}$  qui nous permettent de trouver une bonne approximation de la solution  $|\psi_{best}\rangle$  qui minimise notre Hamiltonien. La figure 4.2 représente l'architecture générale de l'approche VQE.

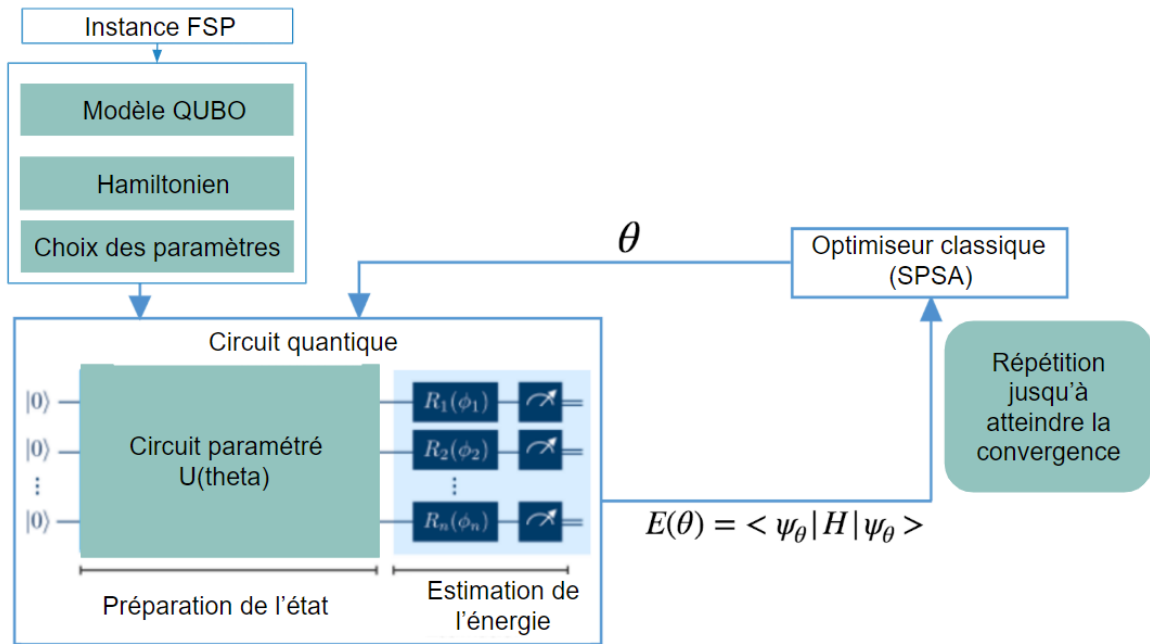


FIGURE 4.2 – Architecture générale de VQE

Dans ce qui suit, nous allons présenter les différents choix de l'optimiseur classique et le circuit paramétré.

#### 4.4.1.1 L'optimiseur classique

Nous avons choisi de travailler avec la méthode d'approximation stochastique des perturbations simultanées (SPSA). Cette méthode est une technique d'optimisation basée sur le gradient, bien adaptée aux problèmes pour lesquels il est difficile ou impossible d'obtenir directement un gradient de la fonction objectif en fonction des paramètres. Cette méthode est alors suggérée lorsqu'une solution sous forme proche du problème n'est pas disponible et lorsque la fonction objectif peut être contaminée par du bruit, ce qui est le cas de nombreuses fonctions objectifs dans les algorithmes variationnels quantiques.

L'idée clé de l'approche SPSA est de fournir une approximation du gradient, en se basant uniquement sur les mesures de la fonction objectif. SPSA utilise une approximation stochastique du gradient, en perturbant simultanément, avec un vecteur spécifique,  $\Delta(k)$  tous les paramètres aléatoirement. Considérons une fonction donnée  $f : R^n \rightarrow R$  à minimiser, où  $n$  est le nombre de variables et le vecteur de paramètres est  $\theta \in R$ . Étant donné un point initial  $\theta^{(0)}$  et un petit taux d'apprentissage  $\eta > 0$ , la mise à jour des paramètres à la  $k$ -ième itération est égale à l'expression suivante :

$$\theta^{(k+1)} = \theta^{(k)} - \eta * \nabla f(\theta^{(k)}) \quad (4.21)$$

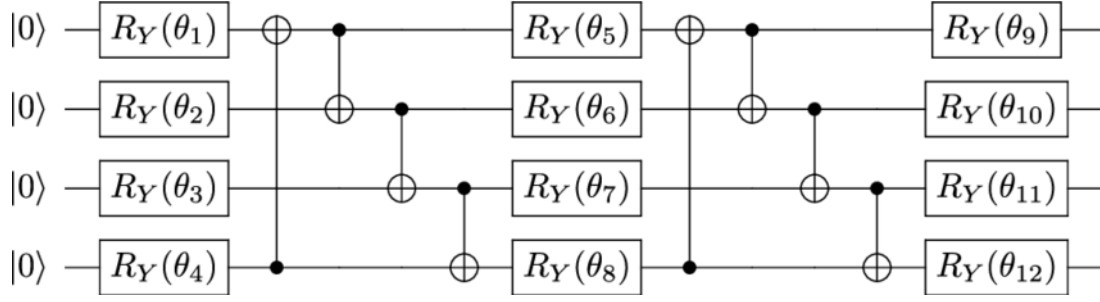
Avec une petite taille de pas d'expansion  $\epsilon$ , le gradient est approximé comme suit :

$$\nabla f(\theta^{(k)}) \approx \frac{(f(\theta^{(k)} + \epsilon * \Delta(k)) - (f(\theta^{(k)} - \epsilon * \Delta(k)))}{2\epsilon} * \Delta(k) \quad (4.22)$$

La condition d'arrêt peut être contrôlée en étudiant comment les paramètres changent pendant la minimisation ou par une approche de force brute avec un nombre maximum donné d'itérations. Dans notre cas, nous avons choisi un nombre maximum d'itérations fixe.

#### 4.4.1.2 Le circuit paramétré

Nous avons choisi de travailler avec un Ansatz générale appelé les deux locales qui est composé d'une alternance de couches de rotation et de couches d'intrication. Les couches de rotation sont des portes à un seul qubit appliquées sur tous les qubits. La couche d'intrication utilise des portes à deux qubits pour créer une relation d'intrication sur les deux qubits. L'avantage avec cette ansatz est qu'il est adapté au matériel quantique disponible aujourd'hui et que la profondeur de ce circuit est relativement petite comparé aux ansatz disponibles.

FIGURE 4.3 – Le circuit paramétré **Two Local**

La figure 4.3 représente le circuit deux locales qui utilisent comme porte quantique de la couche de rotation la porte quantique  $R_y$  et comme portes quantiques pour la couche d'intrication la porte  $CNOT$  présentée dans la section 1. L'idée ici est d'essayer de générer de nouveaux états en introduisant des modifications sur les amplitudes associées à chaque état. L'algorithme de VQE est décrit ci-dessous.

---

**Algorithme 8 : Pseudo-Algorithme VQE**


---

- 1 **procedure** VQE
  - 2 Générer Le modèle qubo
  - 3 Convertir qubo vers un hamiltonien  $H$
  - 4 Choisir l'ansatz du circuit
  - 5 Construire le circuit de l'opérateur  $U(\theta)$
  - 6 Définir l'état initial
  - 7 Générer le circuit VQE
  - 8 Applique le processus d'optimisation classique pour trouver le paramètre  $\theta$
  - 9 Retourner la solution
  - 10 **end procedure**
-



### 4.4.2 Quantum Approximate Optimisation Algorithm

Comme toutes méthodes variationnels, cette méthode cherche à trouver une approximation pour la valeur qui minimise la fonction objectif tel que :

$$f(\gamma, \beta) = \langle \psi(\gamma, \beta) | H | \psi(\gamma, \beta) \rangle \quad (4.23)$$

tel que  $\gamma$  et  $\beta$  représentent les paramètres du circuit quantique que l'optimiseur classique essaie de trouver. La différence entre cette méthode et la méthode VQE réside dans le circuit paramétré utilisé pour la génération de l'état  $\psi(\gamma, \beta)$  qui dépend de notre problème. Le circuit est basé sur deux opérateurs unitaires  $U(H, \gamma)$  appelés l'opérateur de coût,  $U(B, \beta)$  et l'opérateur mixer et un entier  $p \geq 1$ . Les deux unitaires correspondent à des rotations sur nos qubits tandis que l'entier  $p$  désigne la profondeur de notre circuit (Figure 4.4).

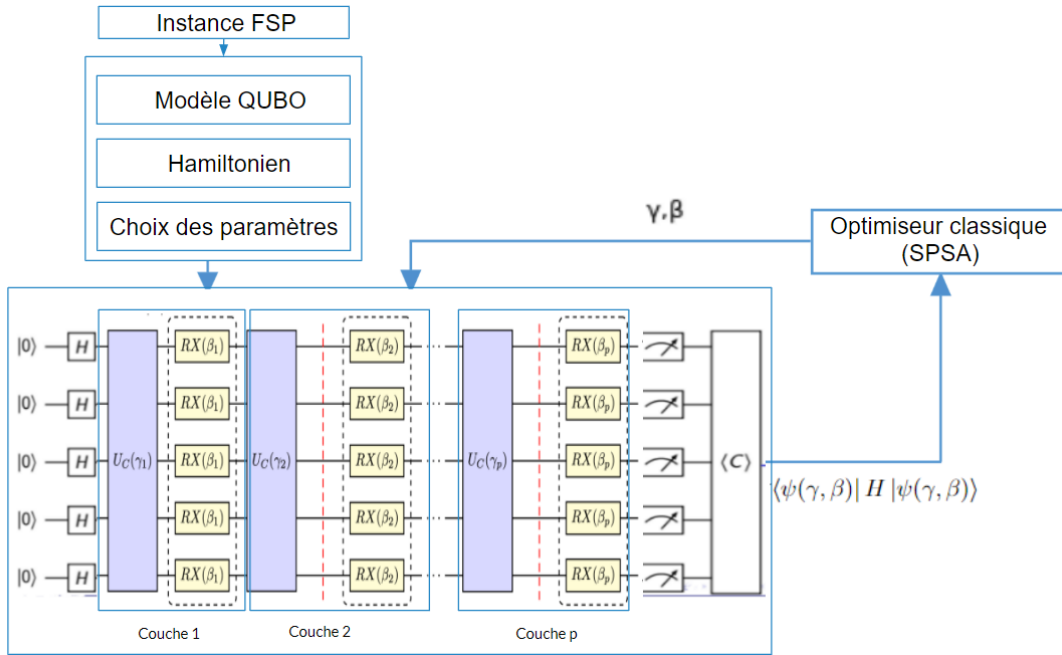


FIGURE 4.4 – Architecture générale de QAOA-FSP

Nous avons choisi comme optimiseur classique l'optimiseur (SPSA) présenté dans la section 4.4.1.1. L'opérateur de coût  $U(H, \gamma)$  est constitué de le Hamiltonien du problème PFSP et un angle  $\gamma$ . il est exprimé par l'équation ci-dessous :

$$U(H, \gamma) = e^{-i\gamma H} \quad (4.24)$$

L'opérateur mixer est utilisé pour mélanger les états afin que tous les états soient testés. Cette opération est réalisée par un mixeur  $M$  et un angle  $\beta$ .

$$U(M, \beta) = e^{-i\beta M} \quad (4.25)$$

Tel que  $M$  est la somme des résultats de l'application de la matrice Pauli sur les qubits :

$$M = \sum_{j=1}^n \sigma_j^x \quad (4.26)$$

Ces portes sont appliquées à l'état initial sur plusieurs couches  $p$  consécutives. Plus le nombre de couches est grand, plus les performances de l'algorithme augmentent, mais ceci nécessite également plus d'opérations sur l'ordinateur quantique. Chaque couche  $i$  possède deux paramètres  $\beta_i$  et  $\gamma_i$ . Au finale, l'expression de l'état obtenu peut être écrite par :

$$|\psi(\gamma, \beta)\rangle = U(M, \beta_p)U(H, \gamma_p)U(M, \beta_{p-1})U(H, \gamma_{p-1})...U(M, \beta_1)U(H, \gamma_1) |\psi_0\rangle \quad (4.27)$$

Tel que l'état initial est égale à la superposition uniforme de tous les états possibles.

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_z |z\rangle \quad (4.28)$$

L'algorithme de l'approche est décrit dans ce qui suit :

---

**Algorithme 9 : Pseudo-Algorithme QAOA**


---

- 1 **procedure** QAOA
  - 2 Générer Le modèle qubo
  - 3 Convertir qubo vers un hamiltonien  $H$
  - 4 Définir le mixer  $M$
  - 5 Construire le circuit des opérateurs  $U(H, \gamma)$  et  $U(M, \beta)$
  - 6 Définir l'état initial
  - 7 Fixer le paramètre  $p$  et générer le circuit qaoa
  - 8 Applique le processus d'optimisation classique pour trouver les paramètre  $\beta$  et  $\gamma$
  - 9 Retourner la solution
  - 10 **end procedure**
- 

### 4.4.3 QAOA-Swap

Dans cette méthode, nous proposons une modification de la partie mixer du circuit de QAOA. La partie mixer de l'algorithme QAOA est basé sur des portes pauli  $X$  1 qui permet de permuter les probabilités pour chaque qubits  $|0\rangle$  et  $|1\rangle$  de l'état. Dans cette approche, nous allons modifier le mixer et utiliser un mixer basé sur les portes SWAP qui permet de permuter entre les qubits au lieu de permuter les probabilités d'apparition. La

matrice swap est exprimée comme suit :

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Afin d'augmenter la probabilité de générer des solutions faisables, nous allons appliquer les swap d'une façon qui nous permet de générer de nouvelles permutations. L'idée de base est d'appliquer les swap entre deux qubits qui représentent la même position. Chaque qubit représente une variable de décision donc un swap est appliqué entre deux qubits qui représentent des variables de décision  $x_{i,p}$  et  $x_{j,p}$ . Notre swap mixer est exprimé comme suit :

$$M = \sum_{i=1}^{N-1} SWAP_{(i,p),(i+1,p)} \quad \forall p \quad (4.29)$$

Par conséquent, notre opérateur mixer est comme suit :

$$U(M, \gamma) = e^{-i\gamma M} \quad (4.30)$$

Il est à noter que la porte SWAP est un opérateur hermitien et unitaire donc, nous pouvons exprimer notre opérateur SWAP comme suit :

$$e^{-i\gamma SWAP} = \begin{bmatrix} \cos(\gamma) + i\sin(\gamma) & 0 & 0 & 0 \\ 0 & \cos(\gamma) & i\sin(\gamma) & 0 \\ 0 & i\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 0 & \cos(\gamma) + i\sin(\gamma) \end{bmatrix}$$

En utilisant notre opérateur swap, pour un exemple de trois tâches, notre mixer est décrit comme suit :

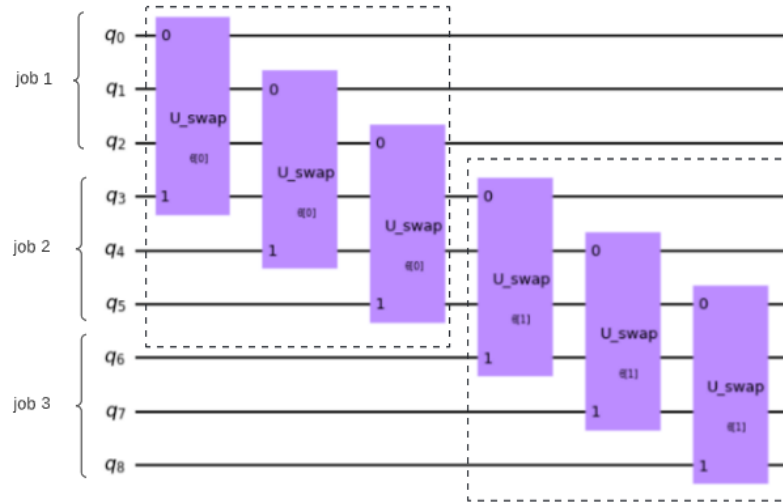


FIGURE 4.5 – Le swap mixer pour une instance de trois jobs

L'algorithme permettant de générer le SWAP mixer est décrit dans ce qui suit :

---

**Algorithme 10 :** Pseudo-Algorithme QAOA Swap
 

---

- 1 **procedure** QAOA SWAP
  - 2 Générer Le modèle qubo
  - 3 Convertir qubo vers un hamiltonien  $H$
  - 4 Construire l'opérateur paramétré SWAP mixer  $U_{swap}$
  - 5 Construire le circuit des opérateurs  $U(H, \gamma)$  et  $U(M, \beta)$
  - 6 Définir l'état initial
  - 7 Fixer le paramètre  $p$  et générer le circuit qaoa
  - 8 Applique le processus d'optimisation classique pour trouver les paramètre  $\beta$  et  $\gamma$
  - 9 Retourner la solution
  - 10 **end procedure**
-

## 4.5 Conclusion

Dans ce chapitre, nous avons présenté un deuxième type d'algorithmes quantiques : les algorithmes hybrides. Nous avons commencé par proposer deux modélisations mathématiques sous la forme QUBO pour le PFSP, la première modélisation consiste en l'approximation du problème vers un problème du voyageur de commerce puis l'utilisation du modèle QUBO proposée pour la résolution de ce problème. Le deuxième modèle proposé consiste en l'utilisation d'une modélisation linéaire proposé pour la résolution du PFSP puis l'approximation de cette modélisation pour avoir un modèle QUBO.

Afin de résoudre le problème étudié, nous avons proposé trois approches, la première consiste en l'utilisation du VQE et la deuxième approche consiste en l'utilisation de la méthode QAOA pour la résolution du PFSP. La dernière approche est une deuxième version du QAOA où nous avons modifié la partie mixer du circuit. Dans le chapitre suivant, nous allons présenter la mise en œuvre de ces approches et leurs résultats.

# Chapitre 5

## Implémentation, Tests et Analyse des performances

Dans cette partie, nous allons présenter les différentes technologies utilisées pour la mise en œuvre de notre solution : langage de programmation, environnement de développement, environnement d'exécution et bibliothèques utilisées. Dans un deuxième lieu, nous allons présenter l'implémentation des phases de l'approche. Enfin, nous présentons les résultats obtenus ainsi qu'une analyse des performances.

### 5.1 Outils

Dans cette section, Nous présentons l'environnement sur lequel les tests ont été menés et les technologies utilisées pour implémenter les différents modules du système.

#### 5.1.1 Langage de programmation

Afin d'implémenter notre solution, nous avons utilisé le langage de programmation Python, Python est un langage de programmation interprété, interactif et orienté objet. Il intègre des modules, des exceptions, un typage dynamique, des types de données dynamiques de très haut niveau et des classes. Il supporte multiples paradigmes de programmation au-delà de la programmation orientée objet, tels que la programmation procédurale et fonctionnelle. Python offre des outils de haut niveau et une syntaxe simple à utiliser, il permet d'optimiser d'une manière remarquable la productivité des programmeurs. Il comporte aussi de nombreuses bibliothèques spécialisées ainsi que des extensions pour plusieurs applications.

#### 5.1.2 Bibliothèques

Pour réaliser cette approche, nous avons utilisé les libraires suivantes :

1. **Qiskit** : Qiskit est un nouveau framework open-source permettant d'effectuer des calculs quantiques. elle fournit une API en langage de programmation Python pour une expérimentation et un apprentissage faciles et simples. elle permet à ses utilisateurs de créer des circuits quantiques et de les visualiser ainsi que d'utiliser des fonctionnalités plus complexes. Qiskit propose également une grande variété d'algorithmes et de fonctions prêts à l'emploi. elle permet de créer des circuits quantiques et de les visualiser. Il est bien intégré à IBM Quantum Experience et les utilisateurs peuvent exécuter leurs programmes sur de véritables ordinateurs quantiques de manière très simple et robuste. Il fournit différents simulateurs, modèles de bruit et algorithmes. Qiskit est constitué de 4 composants principaux :

- (a) **Aer** est le plus bas dans la hiérarchie. Il fournit des simulateurs et des émulateurs.
- (b) **Terra** qui comportent les éléments de base qui permettent la création de circuits quantiques. Il fournit une interface entre l'API de haut niveau de l'utilisateur final et la gestion de la communication et de l'ordonnancement des impulsions du backend.
- (c) **Aqua et Ignis** sont des bibliothèques supplémentaires construites au-dessus de Terra. Aqua est un élément de haut niveau qui fournit des algorithmes et des structures de haut niveau. Ignis est un élément pour l'imitation du bruit.

Nous avons utilisé qiskit pour la construction et l'exécution des circuits quantiques.

2. **Matplotlib** : Matplotlib est une bibliothèque permettant la visualisation de données sous forme graphique. elle est utilisée dans des scripts python pour générer des graphes et des courbes. Nous avons utilisé cette bibliothèque pour visualiser les résultats obtenue après l'exécution des circuits quantiques.
3. **Qiskit-optimization** : est un module proposé par qiskit permettant la modélisation mathématique en utilisant docplex, elle facilite aussi l'implémentation des différentes méthodes variationnelles et l'extension de ces derniers. Des optimiseurs classiques compatibles sont également fournis pour les tests, la validation et l'évaluation comparative.
4. **Docplex** est une bibliothèque destinée à la modélisation de la programmation par contraintes pour Python. et la programmation quadratique, ce qui aide beaucoup et facilite la mise en correspondance avec le modèle d'Ising. Nous avons utilisé cette librairie pour générer notre modèle QUBO et pour la génération de l'hamiltonien.
5. **Streamlit** nous avons utilisé cette bibliothèque pour développer notre interface graphique.

### 5.1.3 Plateforme de développement / IDE

Afin de bien visualiser les résultats, la modélisation des résultats et faciliter la collaboration entre nous, nous avons travaillé avec Jupyter Notebook et GitHub. Ces derniers sont présentés dans ce qui suit :

1. **GitHub** : est un service web permettant la gestion collaborative de développement des logiciels, basé sur le programme Git. Il permet également de sauvegarder l'historique des versions précédentes, par conséquent, il permet de récupérer les versions précédentes et d'introduire des nouvelles fonctionnalités et modifications dans le projet. Dans notre cas, nous avons créé un repository GitHub pour héberger notre code.
2. **Jupyter Notebook** : est une application web open source utilisé pour créer et partager des documents contenant du code, des équations, des visualisations et du texte. les notebooks sont en format de document open-source basé sur JSON qui prend en charge une variété de langages de programmation utilisés par les scientifiques des données, notamment Python et R. Nous avons utilisé Jupyter Notebook pour visualiser et générer les différentes parties du circuit.
3. **Visuel Studio** : Afin d'implémenter notre solution et nos modules, nous avons utilisé visuel studio.

### 5.1.4 Plateforme d'exécution

1. **IBM Quantum Computing** : est une plateforme en ligne permettant un accès public et premium aux services d'informatique quantique en cloud fournis par IBM Quantum. Cela comprend l'accès à un ensemble de processeurs quantiques d'IBM, les simulateurs quantiques, un ensemble de tutoriels sur le calcul quantique et l'accès à un manuel interactif. En février 2021, le service comptait plus de 20 dispositifs, dont six sont librement accessibles au public. Ce service peut être utilisé pour exécuter des algorithmes et des expériences, et explorer des tutoriels et des simulations autour de ce qui pourrait être possible avec l'informatique quantique. Nous avons opté à l'utilisation de cette plateforme pour l'exécution de nos circuits quand leurs profondeurs n'est pas très grandes, car la simulation d'un système quantique sur une simple ordinateurs classique devient impossible quand le nombre de qubits augmente.
2. **Opale** : une machine disponible dans le laboratoire LIST3N avec des performances modeste, mais qui nous a permise d'exécuter nos programmes. en effet, pour l'exécution des circuits de Grover, nous avons utilisé la plateforme IBM Quantum Computing après pour l'exécution des circuits, mais après pour le teste de GAS et de BSG, nous avons rencontré des problèmes de technique lié au temps d'exécution et la



perte de connexion du coup, nous avons choisi d'installer les simulateurs localement sur Opale.

### 5.1.5 Simulateur quantique

Le choix du simulateur quantique représente une étape très critique, car chaque simulateur possède ses avantages et ses inconvénients, soit en termes du temps nécessaire pour la simulation des systèmes quantique, soit en termes du nombre de qubits qu'on peut simuler. Dans notre cas, nous avons choisi de travailler avec le simulateur de qiskit **MPS**. Ce dernier nous permet de dépasser la frontière de 32 qubits imposée par QASM et de simuler des systèmes avec 100 qubits.

## 5.2 Implémentation des approches proposés

Notre solution permet de résoudre le problème du PFSP avec des méthodes quantique. Deux types de méthodes ont été proposées : les méthodes variationnels et les méthodes basée sur l'algorithme de Grover. Nous avons implémenté notre solution en suivant une programmation orientée objet. La figure 5.1 présente l'ensemble des classes implémentées.

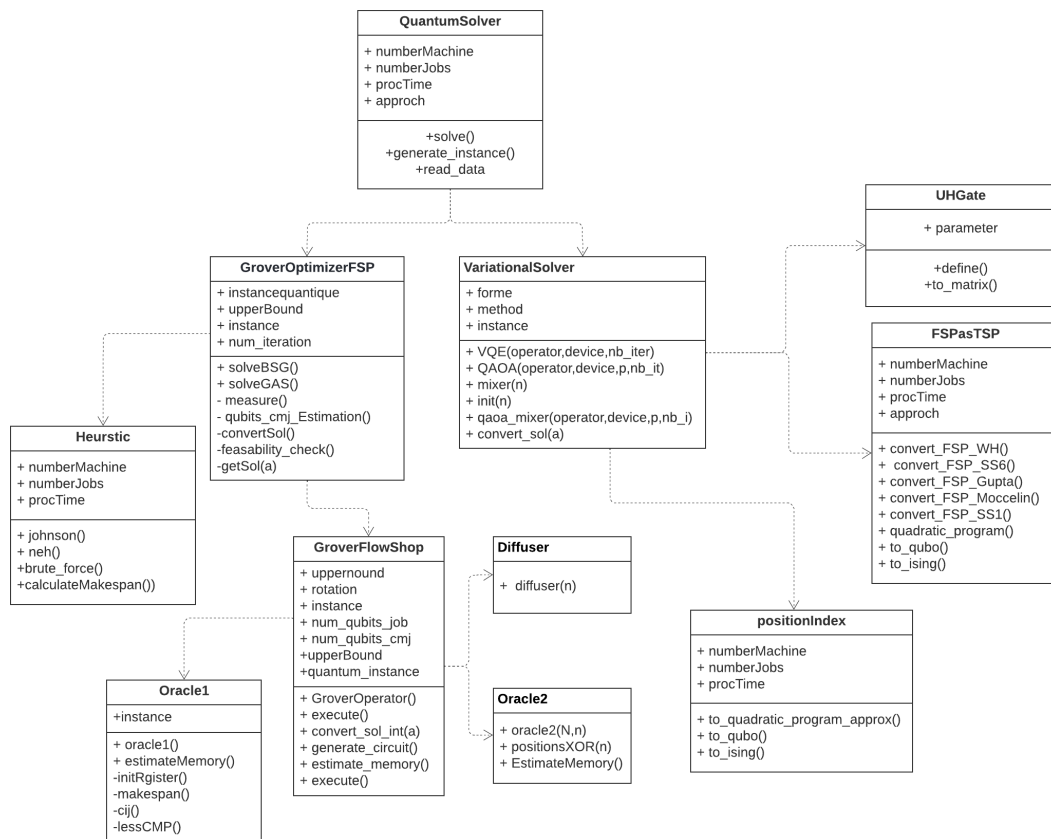


FIGURE 5.1 – Diagramme de classe

### 5.3 Les benchmarks de tests

Les limites liées aux machines quantiques nous ont amenés à utiliser des simulateurs quantiques pour exécuter nos circuits quantiques. Comme la simulation quantique est un problème NP-difficile, nous n'avons pas pu faire des tests sur de grandes instances et nous nous sommes limités à des instances avec quatre tâches et deux machines pour l'algorithme de Grover. Ce type d'instances est considéré comme facile pour les algorithmes classiques. Pour les algorithmes variationnels, nous nous sommes limités aux instances à de 4 tâches.

Afin de générer ces instances, nous avons implémenté un générateur d'instance basé sur la loi uniforme. Dans le cas de l'algorithme de Grover, les instances générées possèdent des temps d'exécution qui ne dépassent pas quatre car le nombre de qubits nécessaire pour implémenter le circuit augmente exponentiellement avec la taille de l'instance.

### 5.4 Analyse des performances de Grover-FSP

La phase d'implémentation de la solution passe par deux phases principales, l'implémentation d'un générateur du circuit de Grover, puis l'implémentation des méthodes GAS et BSG qui exploitent le circuit de Grover.

#### 5.4.1 Génération du circuit de Grover

Afin d'automatiser la tâche de la construction du circuit de Grover, nous avons implémenté un générateur du circuit comme décrit dans la figure 5.2.

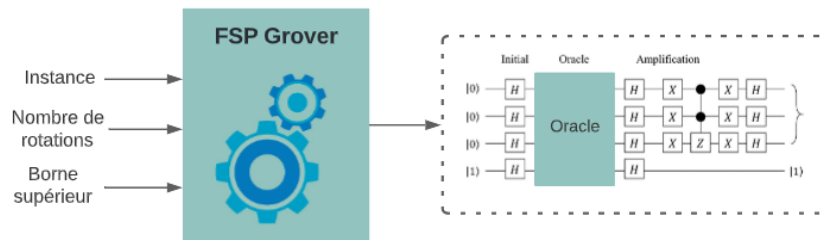


FIGURE 5.2 – Générateur du circuit de Grover

La construction du circuit de Grover passe par trois étapes principales, l'ajout de la partie initialisation, l'ajout de l'oracle et du circuit de diffusion. Dans ce qui suit, nous allons présenter une illustration du circuit de Grover généré ainsi que la réalisation de l'opérateur oracle (la partie de circuit qui représente notre contribution) et l'opérateur de diffusion. Nous avons choisi comme exemple illustratif une petite instance de deux jobs et deux machines. Les temps d'exécution de chaque job sur chaque machine est décrit dans le tableau suivant :

	M1	M2
J1	1 (01)	2 (10)
J2	2 (10)	3 (11)

La sortie finale du générateur est le circuit complet de Grover. La figure 5.3 présente le circuit de Grover généré pour un nombre de rotations égale à 2.

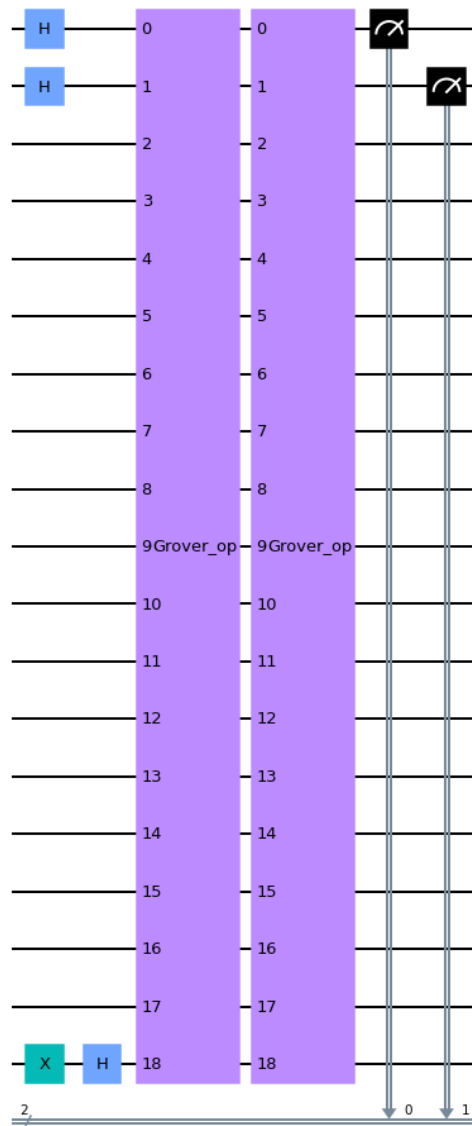


FIGURE 5.3 – Circuit de Grover généré

L'opérateur Grover-op se compose des deux parties oracle et diffusion. La construction de cet opérateur passe par deux étapes : la première étape consiste en la construction de la partie oracle 1 qui nous permet de calculer le  $C_{max}$  et de le comparer avec la borne supérieure. La deuxième partie consiste en la vérification de la validité des ordonnancements.

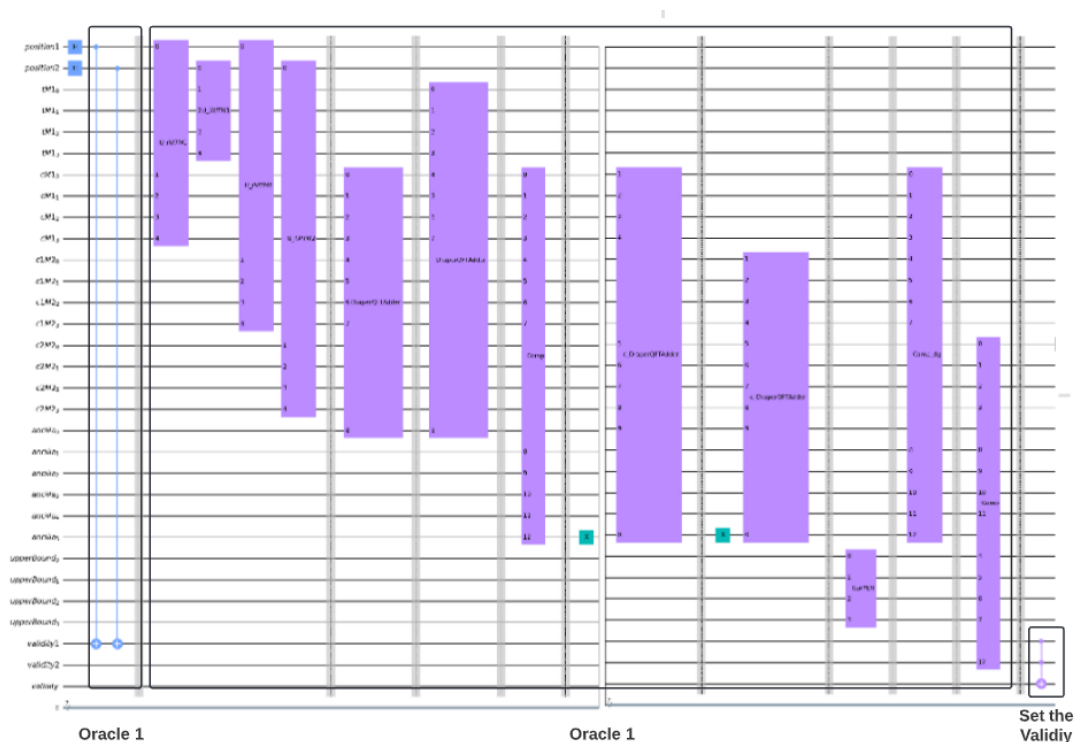


FIGURE 5.4 – Oracle généré pour deux tâches

### 5.4.2 Tests du circuit du Grover

Dans ce qui suit, nous allons présenter les résultats des tests du circuit de Grover. Les tests ont été menés sur trois instances. En initialisant la borne supérieures avec l'algorithme de Johnson. Il est à noter que le problème du PFSP à deux machines n'est un problème NP-difficile, mais le problème de trouver toutes les solutions reste un problème NP-difficile.

#### 5.4.2.1 Test 1

- Instance :  $[[3, 1, 4, 1], [1, 3, 1, 2]]$
- Nombre de solutions attendues : 1
- Nombre de rotations optimal : 3.97
- Solution attendue :  $[0, 1, 3, 2], [0, 3, 1, 2], [1, 0, 3, 2], [1, 2, 3, 0], [1, 3, 0, 2], [1, 3, 2, 0], [3, 0, 1, 2], [3, 1, 0, 2], [3, 1, 2, 0], [3, 2, 1, 0]$
- **Solution trouvée**

L'histogramme présente la fréquence d'apparition des états quantiques après application de la mesure. Nous avons lancé le programme pour un nombre de shots égale à 1000.

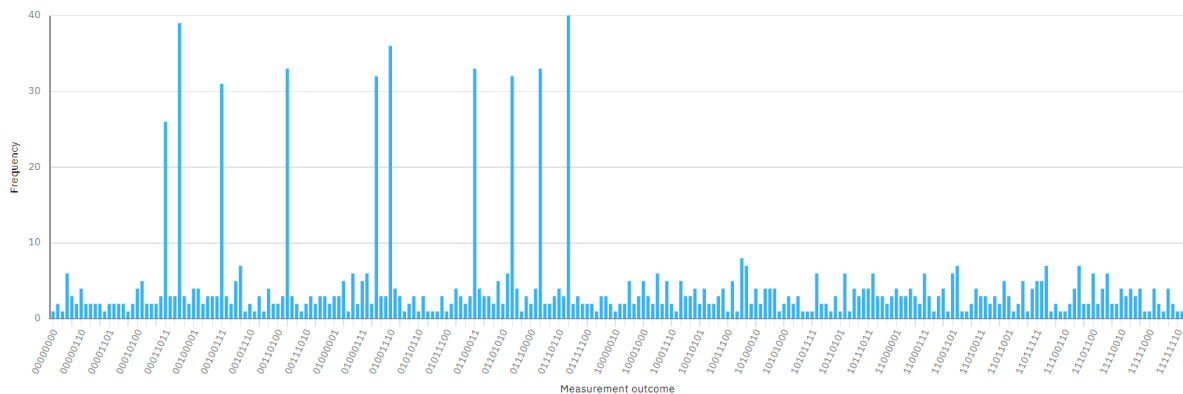


FIGURE 5.5 – Test 1 avec le nombre de rotations égale à 1

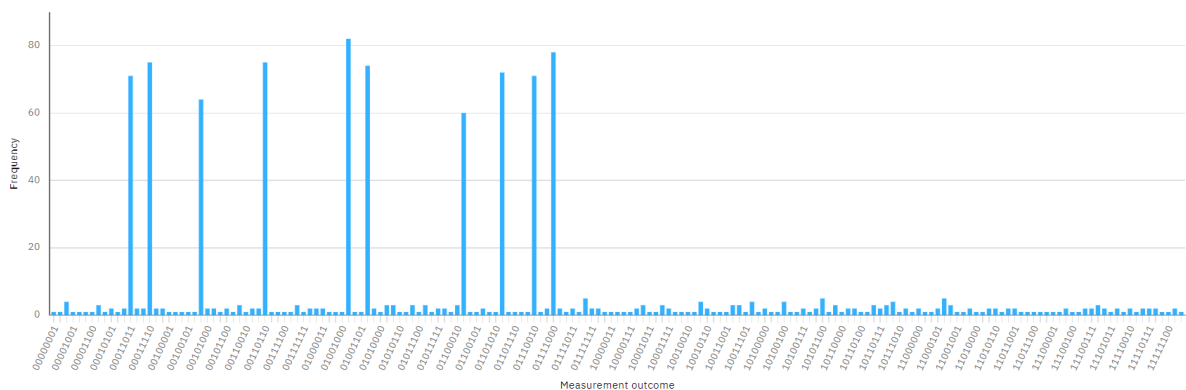


FIGURE 5.6 – Test 1 avec le nombre de rotations égale à 2

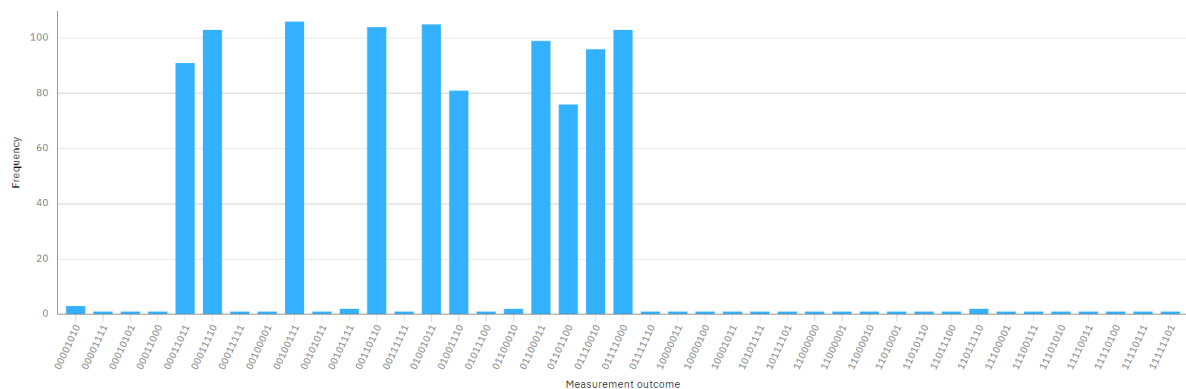


FIGURE 5.7 – Test 1 avec le nombre de rotations égale à 3

d'après les histogrammes présentés ci-dessus, nous pouvons remarquer que les permutations : 11011000 (3,1,2,0), 01111000 (1,3,2,0), 11100100 (3,2,1,0), 01101100 (1,2,3,0), 11010010 (3,1,0,2), 11000110 (3,0,1,2), 00110110 (0,3,1,2), 01001110 (1,0,3,2), 00011110 (0,1,3,2) possèdent le nombre de fréquences d'apparition le plus grand.

D'après les résultats obtenus, nous pouvons déduire que la fréquence d'apparition des solutions est plus grande par rapport aux autres états et plus le nombre de rotations est

proche du nombre de rotations optimales plus la probabilité d'apparition des états qui ne représente pas des solutions est réduite.

#### 5.4.2.2 Test 2

- Instance :  $[[3, 3, 2, 1], [2, 4, 1, 1]]$
- Nombre de solutions attendues : 8
- Nombre de rotations optimal : 4
- Solution attendue :  $[1, 0, 2, 3], [1, 0, 3, 2], [1, 2, 0, 3], [1, 2, 3, 0], [1, 3, 0, 2], [1, 3, 2, 0], [3, 1, 0, 2], [3, 1, 2, 0]$
- Solution trouvée :

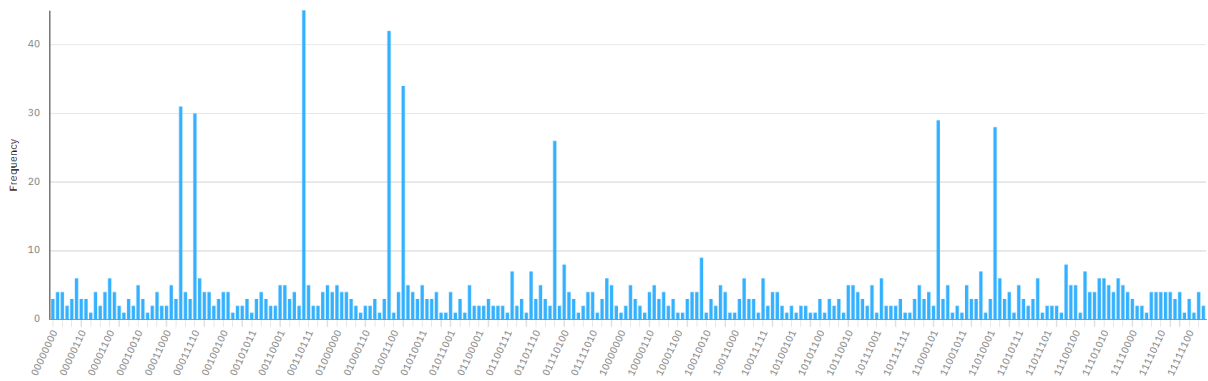


FIGURE 5.8 – Test 2 avec le nombre de rotations égale à 1

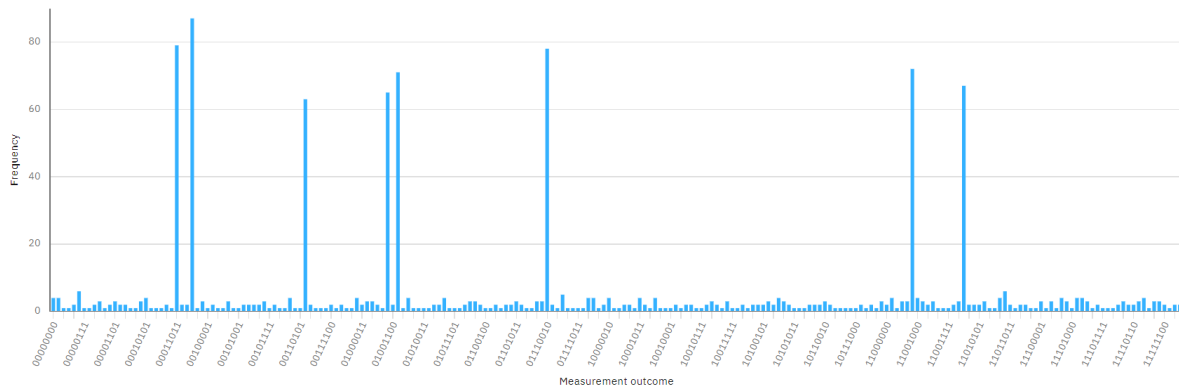


FIGURE 5.9 – Test 2 avec le nombre de rotations égale à 2

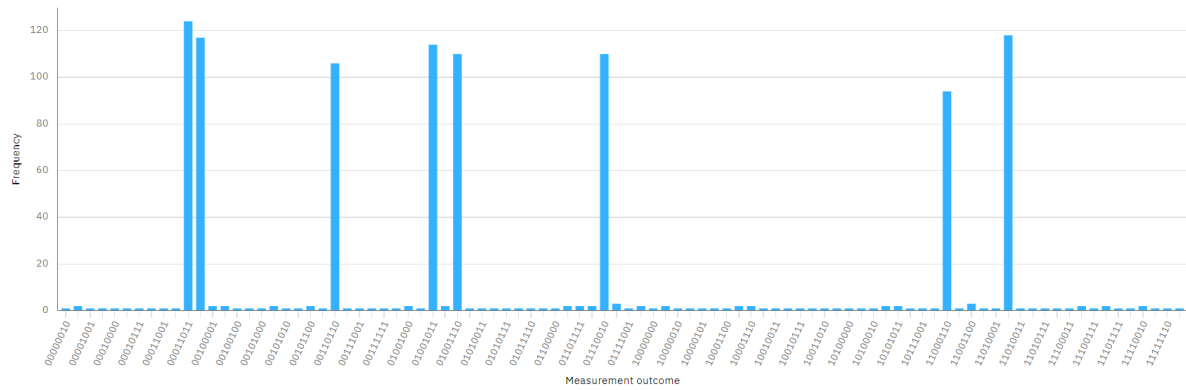


FIGURE 5.10 – Test 2 avec le nombre de rotations égale à 3

D'après les résultats obtenus, nous pouvons déduire que plus le nombre de rotations est grand, plus la fréquence d'apparition de ces résultats augmentent et même, il y avait des états qui ne représentent pas des solutions avec une fréquence d'apparition égale à 0.

#### 5.4.2.3 Test 3

- Instance :  $[[4,2,2,1],[2,3,1,3]]$
- Nombre de solutions attendues : 1
- Nombre de rotations optimal : 12
- Solution attendue :  $[3, 1, 0, 2]$
- Solution trouvée :

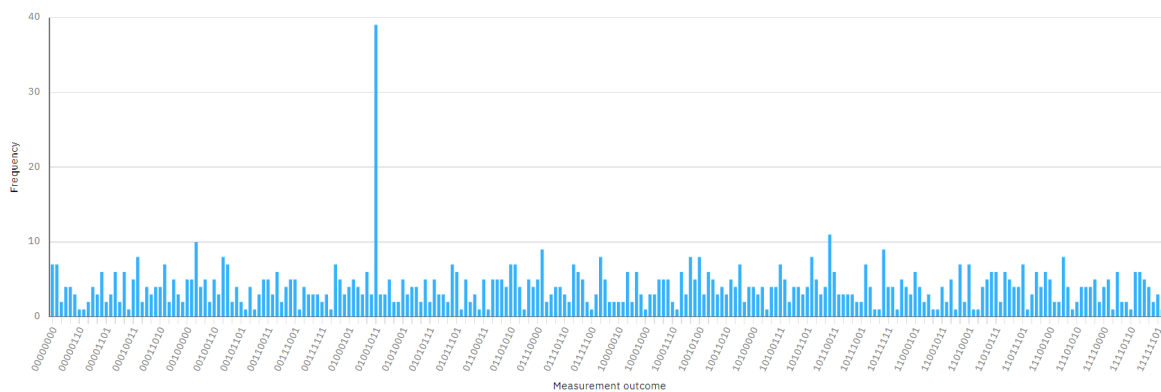


FIGURE 5.11 – Test 3 avec le nombre de rotations égale à 1

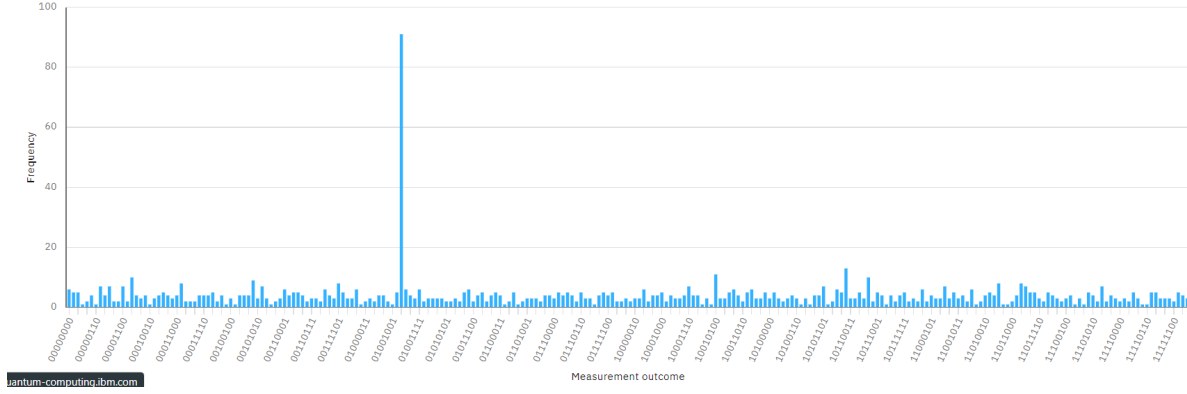


FIGURE 5.12 – Test 3 avec le nombre de rotations égale à 2

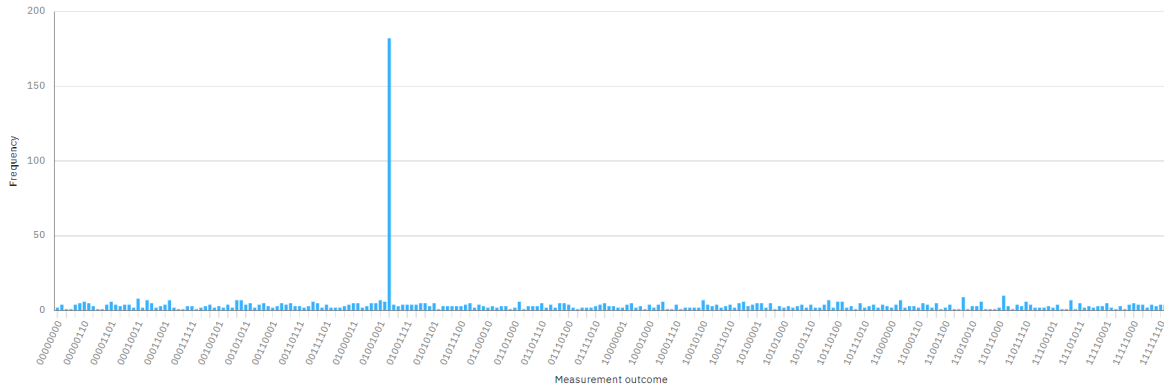


FIGURE 5.13 – Test 3 avec le nombre de rotations égale à 3

### 5.4.3 Analyse des performances du circuit de Grover

Pour bien comprendre les capacités actuelles et futures de cet algorithme, nous analysons le coût quantique de notre circuit GroverFSP. Les deux facteurs les plus importants pour nous à l'ère NISQ sont le nombre de qubits requis et la profondeur du circuit. Dans notre cas, nous avons choisi de nous focaliser sur le premier facteur.

Le nombre de qubits représentent un critère d'évaluation très important du fait que les ordinateurs quantiques disponibles aujourd'hui disposent de centaines de qubits seulement. Dans cette partie, nous allons étudier la scalabilité du nombre de qubits avec la taille de l'instance. Dans un premier lieu, nous avons étudié la corrélation avec le nombre de tâches  $N$ . Le graphe 5.14 montre que le nombre de qubits nécessaires pour implémenter le circuit augmente exponentiellement avec  $N$ .



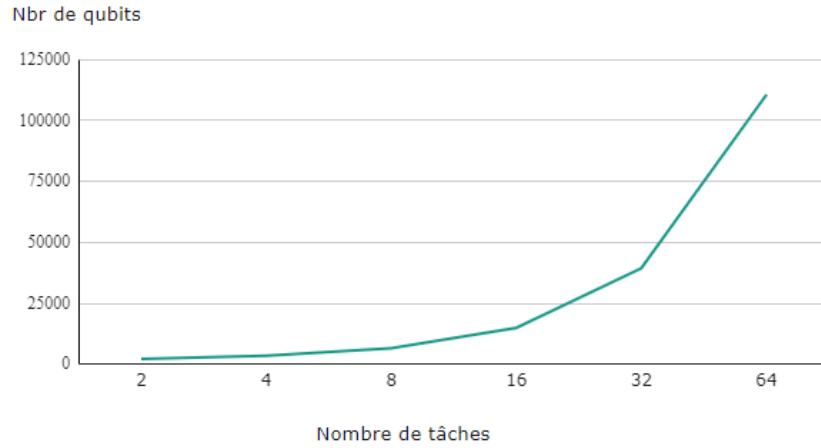
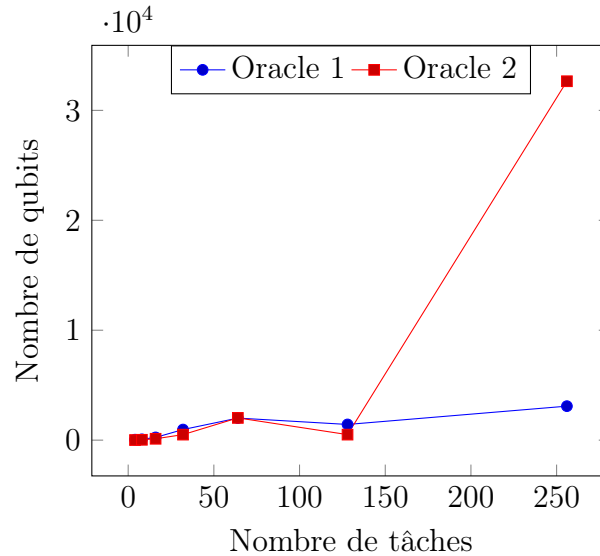


FIGURE 5.14 – Nombre de qubits en fonction de nombre de tâches

Nous avons aussi étudié l'évolution du nombre de qubits nécessaires pour implémenter chaque oracle (oracle 1 et oracle 2) en fonction de nombre de tâches.



Nombre de qubits nécessaires pour implémenter l'oracle 1 et l'oracle 2

Nous pouvons constater d'après la figure précédente que plus le nombre de tâches augmentent, plus le nombre de qubits nécessaires pour implémenter le deuxième oracle augmentent et le pourcentage de nombre de qubits nécessaires pour la réalisation du deuxième oracle devient plus grand par rapport au premier. Nous pouvons en déduire que le nombre de qubits nécessaires pour réaliser le circuit revient à réduire le nombre de qubits nécessaires pour implémenter l'oracle 2. La solution qui semble être la plus performante dans ce cas est de trouver une matrice permettant de générer les solutions faisables, et nous n'aurons pas besoin à ce moment de la partie oracle 2 pour pouvoir distinguer les solutions valides.

Le tableau 5.1 présenté les résultats obtenus après l'exécution des algorithmes GAS et BSG en termes de  $C_{max}$ , nombre d'itérations de l'algorithme et le nombre de lancements de l'algorithme de Grover.

TABLE 5.1 – Résultats de l'exécution des algorithmes GAS et BSG

$N$	<i>GAS</i>			<i>BSG</i>		
	$C_{max}$	$nbiteration$	$nbrlancement$	$C_{max}$	$nbiteration$	$nbrlancement$
2	7	2	12	7	3	8
	8	3	6	8	2	6
	8	3	6	8	3	10
4	16	3	12	16	3	17
	14	4	24	14	3	23

Globalement, nous déduisons que les deux algorithmes nécessitent un nombre très grand de lancement de l'algorithme de Grover pour pouvoir trouver la solution optimale et chaque lancement de Grover nécessite un temps d'exécution important qui dépend de la profondeur de circuit. Certainement les deux méthodes réussissent à atteindre l'optimum global, mais à cause de la contrainte matérielle, nous ne pouvons pas étudier les performances de ces méthodes en termes de temps d'exécution.

## 5.5 Analyse des performances de VQE-FSP

Dans cette partie, nous allons présenter les différents résultats obtenus après l'exécution des différentes méthodes, ainsi qu'une comparaison entre ces méthodes. Il est à noter que ces tests ont été effectués sur des instances de petite taille qui ne dépassent pas les 6 jobs à cause des limites du matériel utilisé. En effet, nous avons utilisé un simulateur pour exécuter nos circuits et plus la taille de l'instance est grande, plus le temps d'exécution est long. Les limites matérielles nous ont amené à proposer ce plan de tests :

1. VQE : 6 Jobs et 6 machines au maximum
2. QAOA : 4 Jobs et 5 machines au maximum avec  $p = 1$ .
3. QAOA swap : 4 Jobs et 5 machines au maximum avec  $p = 1$ .

### 5.5.1 Comparaison entre les différents modèles QUBO proposés

Dans cette partie, nous allons comparer les différentes modélisations mathématiques proposées ( $QUBO_1, QUBO_2, QUBO_3, QUBO_4, QUBO_5, QUBO_6$ ). En effet, les cinq premiers QUBO sont issus de l'approximation du TSP vers FSP en utilisant respectivement les approximations (approche 1, approche 2, approche 3, approche 4, approche 5) présentées dans le tableau 4.1.

Les résultats de l'exécution des différentes méthodes présentées dans les figures 5.15 et 5.17 montrent que le modèle  $QUBO_2$  donne les meilleurs résultats dans le cadre des méthodes VQE et QAOA pour la plupart des instances. Les diagrammes à boîtes montrent

que pour ce modèle 75% des résultats possèdent une déviation inférieure à 25%. Par contre, la modélisation  $QUBO_4$  est meilleure dans le cadre de la méthode  $QAOA_{SWAP}$  comme le montre la figure 5.16. D'après le diagramme à boîtes, 75% des résultats obtenues possèdent une déviation qui ne dépasse pas le 26%.

$N$	$M$	$QUBO1$ <i>Deviation</i>	$QUBO2$ <i>Deviation</i>	$QUBO3$ <i>Deviation</i>	$QUBO4$ <i>Deviation</i>	$QUBO5$ <i>Deviation</i>	$QUBO6$ <i>Deviation</i>
3	3	25,17%	0,00%	25,1%	1,68%	7,38%	24,83%
	3	11,54%	0,00%	19,58%	10,14%	0,00%	31,12%
	4	0,00%	0,00%	0,00%	0,00%	38,49%	13,31%
	4	0,00%	9,75%	19,49%	19,49%	32,13%	19,49%
	5	0,00%	0,49%	9,25%	0,00%	16,30%	9,73%
	5	0,76%	2,03%	5,82%	2,03%	12,15%	2,03%
	6	0,86%	0,00%	8,02%	0,00%	8,02%	26,65%
	6	17,10%	17,10%	17,10%	17,10%	17,10%	32,90%
	7	8,45%	0,00%	2,36%	0,00%	2,36%	2,36%
	7	8,95%	5,68%	7,21%	0,22%	5,68%	5,68%
4	3	0,00%	6,62%	0,00%	16,72%	6,62%	0,00%
	3	30,69%	4,48%	4,48%	11,03%	11,03%	11,03%
	4	38,00%	28,86%	16,57%	28,86%	18,57%	38,00%
	4	5,49%	4,57%	6,10%	5,49%	6,10%	14,33%
	5	20,33%	0,00%	3,02%	26,92%	3,02%	20,33%
	5	28,92%	19,76%	2,17%	2,17%	28,92%	0,00%
	6	20,68%	<i>NF%</i>	10,23%	19,32%	10,91%	30,68%
	6	17,03%	24,25%	0,80%	5,41%	5,41%	6,21%
	7	41,71%	9,11%	3,37%	17,03%	5,74%	3,37%
	7	17,03%	9,11%	3,37%	17,03%	5,74%	3,37%
5	4	22,36%	20,85%	18,09%	37,94	28,14%	33,81%
	4	24,62%	8,54%	17,09%	9,80%	15,08%	35,34%
	5	32,66%	37,44%	34,92%	29,65%	32,66%	17%
	5	2,01%	11,56%	7,04%	11,56%	11,56%	9,96%
	6	15,33%	13,57%	2,76%	5,28%	1,26%	6,10%
	7	47,99%	48,74%	53,02%	54,27%	17,59%	<i>NF%</i>
	7	4,52%	4,02%	17,34%	11,81%	15,33%	10,68%
6	4	33,42%	16,58%	12,56%	12,56%	20,10%	<i>NF%</i>
	5	37,94%	14,82%	<i>NF%</i>	36,18%	<i>NF%</i>	<i>NF%</i>
	6	15,83%	<i>NF%</i>	16,08%	25,38%	28,64%	<i>NF%</i>

TABLE 5.2 – Résultats obtenus avec l'approche VQE avec les différentes approches 1-6

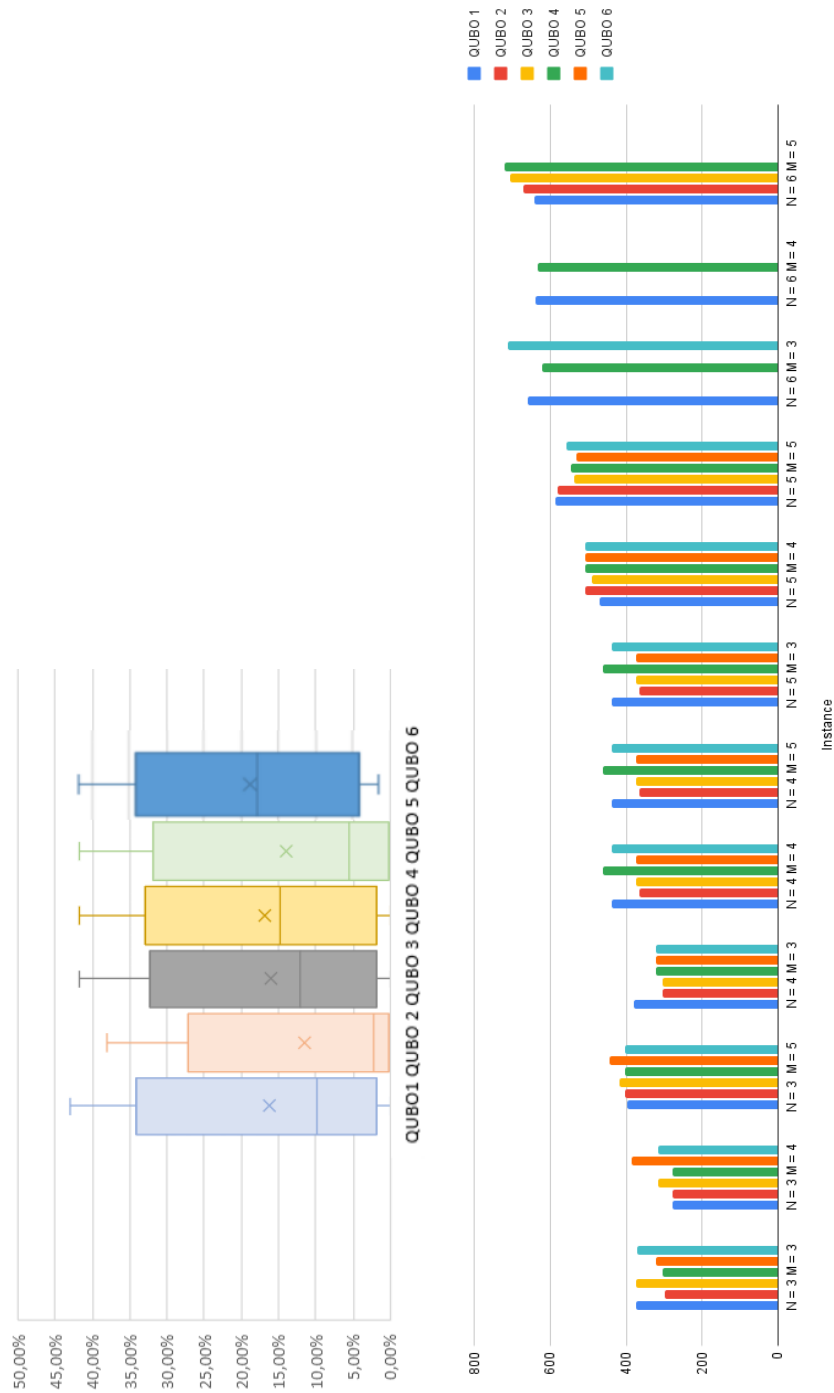


FIGURE 5.15 – Résultats obtenus après l'exécution de l'approche VQE avec les différents modèles QUBO 1-6

TABLE 5.3 – Résultats obtenus après l'exécution de l'approche QAOA avec les différentes approches 1-6

Instance	QUBO 1			QUBO 2			QUBO 3			QUBO 4			QUBO 5			QUBO 6		
	QAOA	QAOA SWAP	QAOA	QAOA	QAOA SWAP	QAOA	QAOA	QAOA SWAP	QAOA	QAOA	QAOA SWAP	QAOA	QAOA	QAOA SWAP	QAOA	QAOA	QAOA SWAP	QAOA
N = 3 M = 3	25,17%	7,38%	1,68%	22,02%	0,00%	0,00%	0,00%	25,17%	0,00%	0,00%	25,17%	0,00%	0,00%	25,17%	17,79%	25,17%	25,17%	25,17%
N = 3 M = 3	0,00%	22,02%	22,02%	26,19%	26,19%	26,19%	26,19%	22,02%	26,19%	26,19%	1,79%	1,79%	0,00%	22,02%	32,14%	22,02%	26,19%	26,19%
N = 3 M = 3	10,14%	0,00%	0,00%	0,00%	19,58%	19,58%	0,00%	31,12%	0,00%	0,00%	31,12%	0,00%	0,00%	31,12%	11,54%	31,12%	31,12%	31,12%
N = 3 M = 4	14,75%	20,50%	14,75%	0,00%	20,50%	20,50%	14,75%	0,00%	13,31%	13,31%	13,31%	13,31%	20,50%	38,49%	21,94%	38,49%	38,49%	38,49%
N = 3 M = 4	0,00%	19,49%	0,00%	0,00%	9,75%	19,86%	19,86%	19,86%	2,53%	19,86%	19,86%	19,86%	19,86%	9,75%	2,53%	19,86%	19,86%	19,86%
N = 3 M = 4	3,88%	27,83%	23,95%	19,74%	3,88%	3,88%	3,88%	0,00%	0,00%	0,00%	27,83%	23,95%	3,88%	3,88%	23,95%	3,88%	3,88%	3,88%
N = 3 M = 5	4,19%	4,19%	2,26%	4,19%	4,19%	10,97%	10,97%	4,19%	10,97%	10,97%	0,00%	4,19%	10,97%	10,97%	2,26%	0,00%	0,00%	0,00%
N = 3 M = 5	0,00%	0,76%	0,76%	12,41%	12,41%	26,08%	26,08%	2,03%	12,15%	12,15%	0,76%	0,76%	0,76%	0,76%	2,03%	2,03%	2,03%	2,03%
N = 3 M = 5	9,73%	9,25%	16,30%	0,00%	0,00%	9,73%	9,73%	9,25%	16,30%	16,30%	9,25%	9,73%	0,00%	0,00%	16,30%	0,00%	0,00%	0,00%
N = 4 M = 3	16,72%	1,58%	16,72%	23,66%	23,66%	1,58%	1,58%	17,67%	16,72%	16,72%	1,58%	NF	17,67%	17,67%	1,58%	13,56%	13,56%	13,56%
N = 4 M = 3	6,55%	36,90%	6,55%	15,17%	15,17%	24,14%	24,14%	25,17%	24,14%	24,14%	8,97%	24,14%	10,34%	10,34%	6,55%	30,69%	30,69%	30,69%
N = 4 M = 3	28,57%	53,28%	0,00%	0,00%	34,36%	34,36%	0,00%	33,98%	28,57%	28,57%	32,05%	28,57%	34,36%	34,36%	NF	34,36%	34,36%	34,36%
N = 4 M = 4	5,49%	7,93%	7,93%	29,88%	29,88%	5,49%	5,49%	24,39%	7,32%	7,32%	26,52%	5,49%	20,43%	20,43%	NF	26,52%	26,52%	26,52%
N = 4 M = 4	38,00%	16,00%	38,00%	38,00%	2,00%	38,00%	38,00%	21,14%	38,00%	38,00%	8,29%	38,00%	14,00%	14,00%	38,00%	8,29%	8,29%	8,29%
N = 4 M = 4	13,17%	26,95%	7,19%	28,44%	28,44%	13,17%	13,17%	44,31%	NF	NF	28,44%	13,17%	25,75%	25,75%	7,19%	45,81%	45,81%	45,81%
N = 4 M = 5	2,17%	14,22%	1,20%	25,54%	25,54%	2,17%	2,17%	25,54%	23,86%	23,86%	16,39%	2,17%	31,81%	31,81%	19,76%	29,40%	29,40%	29,40%
N = 4 M = 5	0,00%	23,97%	0,00%	12,89%	12,89%	22,16%	22,16%	23,97%	NF	NF	22,42%	0,00%	29,90%	29,90%	NF	22,42%	22,42%	22,42%

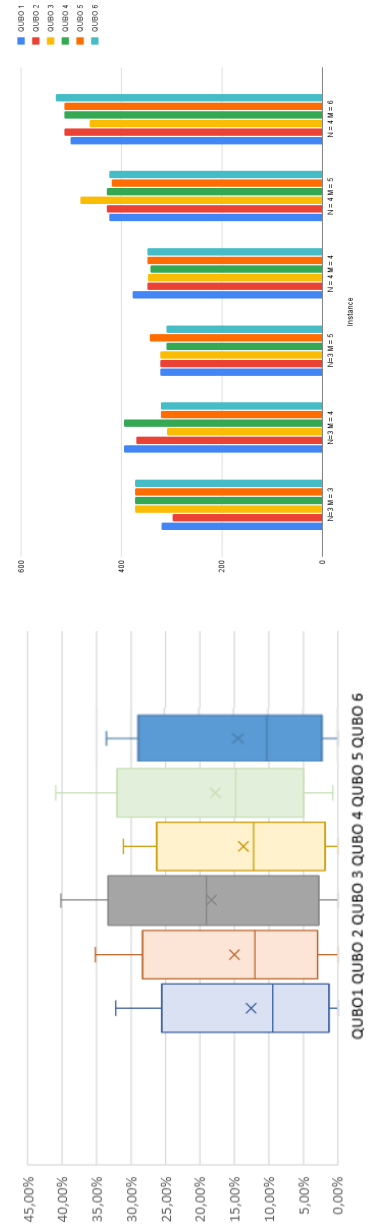


FIGURE 5.16 – Résultats obtenus après l'exécution de l'approche QAOA swap avec les différents modèles QUBO 1-6

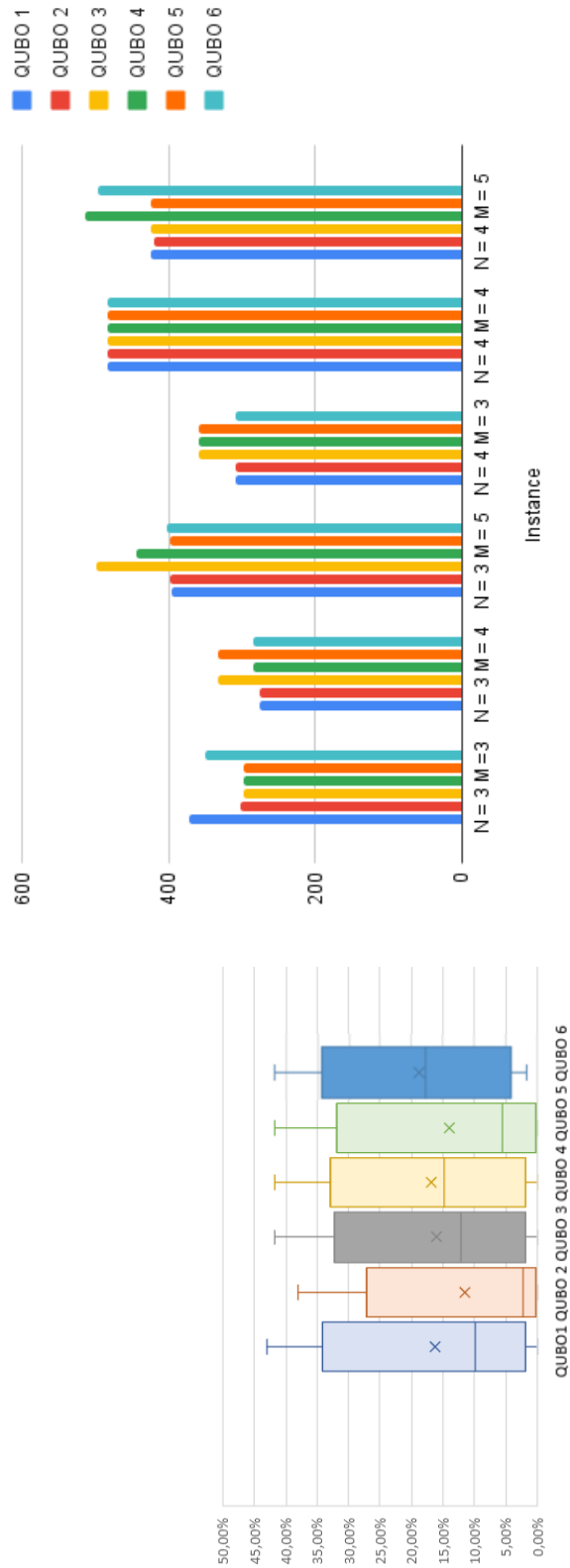


FIGURE 5.17 – Résultats obtenus après l'exécution de l'approche QAOA avec les différents modèles QUBO 1-6

### 5.5.2 Comparaison entre les différentes méthodes implémentées

Dans cette partie, nous allons présenter les résultats obtenus après l'exécution des différentes méthodes implémentées. Nous pouvons comparer les trois méthodes selon trois critères : la qualité de la solution, la faisabilité de la solution et le temps d'exécution. Commenant par la qualité de la solution, d'après les résultats présentés dans le tableau et la figure 5.19, nous remarquons que les méthodes QAOA et VQE donnent de meilleures dérivations par rapport à QAOA SWAP. Nous remarquons également que la méthode QAOA SWAP nous a permis d'éviter les solutions infaisables contrairement à la méthode QAOA (les carrés rouges dans le tableau représentent des solutions non faisables).

En termes de temps d'exécution, nous pouvons dire que la méthode VQE possède le meilleur temps d'exécution par rapport aux méthodes QAOA et QAOA SWAP. Nous déduisons aussi que la version QAOA SWAP représente une amélioration du QAOA en termes de temps d'exécution.

D'une manière globale, nous pouvons dire que la méthode VQE représente le meilleur compromis entre le temps d'exécution et la qualité de la solution par rapport aux méthodes QAOA et QAOA SWAP.

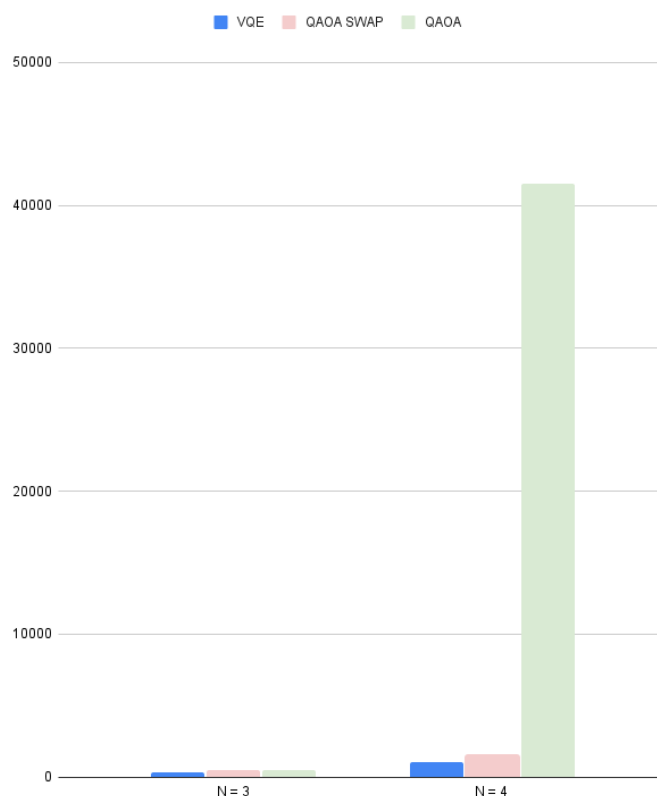


FIGURE 5.18 – Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de temps d'exécution

Instance	QUBO 1			QUBO 2			QUBO 3			QUBO 4			QUBO 5			QUBO 6		
	QAOA	VQE	QAOA SWAP	QAOA	VQE	QAOA SWAP	QAOA	VQE	QAOA SWAP	QAOA	VQE	QAOA SWAP	QAOA	VQE	QAOA SWAP	QAOA	VQE	QAOA SWAP
N = 3 M = 3	25.17%	25.17%	7.38%	1.68%	0.00%	0.00%	25.17%	25.17%	25.17%	0.00%	1.68%	25.17%	0.00%	7.38%	25.17%	17.79%	24.83%	25.17%
N = 3 M = 3	0.00%	0.00%	22.02%	26.19%	22.02%	26.19%	22.02%	22.02%	22.02%	26.19%	1.79%	1.79%	0.00%	1.79%	22.02%	32.14%	22.02%	26.19%
N = 4 M = 3	10.14%	11.54%	0.00%	0.00%	0.00%	19.58%	0.00%	19.58%	31.12%	0.00%	10.14%	31.12%	0.00%	0.00%	31.12%	11.54%	31.12%	31.12%
N = 3 M = 4	14.75%	0.00%	20.50%	14.75%	0.00%	13.31%	0.00%	13.31%	0.00%	13.31%	0.00%	13.31%	20.50%	38.49%	38.49%	21.94%	13.31%	38.49%
N = 4 M = 4	0.00%	0.00%	19.49%	0.00%	9.75%	19.86%	19.49%	19.86%	19.86%	19.86%	19.49%	19.86%	19.86%	32.13%	9.75%	2.53%	19.49%	19.86%
N = 3 M = 4	3.88%	19.74%	27.83%	23.95%	27.83%	19.74%	3.88%	19.74%	19.74%	27.83%	19.74%	27.83%	23.95%	0.00%	3.88%	23.95%	10.36%	3.88%
N = 4 M = 5	4.19%	10.97%	4.19%	2.26%	10.97%	4.19%	4.19%	10.97%	10.97%	10.97%	10.97%	10.97%	4.19%	10.97%	10.97%	2.26%	10.97%	0.00%
N = 3 M = 5	0.00%	0.76%	0.76%	0.76%	2.03%	12.41%	26.08%	5.82%	2.03%	12.15%	2.03%	0.76%	0.76%	12.15%	0.76%	2.03%	2.03%	2.03%
N = 4 M = 5	9.73%	0.00%	9.25%	16.30%	0.49%	0.00%	9.73%	9.25%	9.25%	16.30%	0.00%	9.25%	9.73%	16.30%	0.00%	16.30%	9.73%	0.00%
N = 4 M = 3	16.72%	0.00%	1.58%	16.72%	6.62%	23.66%	1.58%	0.00%	0.00%	16.72%	16.72%	1.58%	24.14%	6.62%	17.67%	1.58%	0.00%	13.56%
N = 4 M = 3	6.55%	30.69%	36.90%	6.55%	4.48%	15.17%	24.14%	4.48%	25.17%	24.14%	11.03%	8.97%	28.57%	11.03%	10.34%	6.55%	11.03%	30.69%
N = 4 M = 3	28.57%	28.57%	53.28%	0.00%	15.83%	34.36%	0.00%	6.56%	33.98%	28.57%	11.97%	32.05%	28.57%	28.57%	34.36%	28.57%	23.55%	34.36%
N = 4 M = 4	5.49%	5.49%	7.93%	4.57%	29.88%	2.00%	29.88%	5.49%	24.39%	7.32%	5.49%	26.52%	5.49%	6.10%	20.43%	14.33%	26.52%	26.52%
N = 4 M = 4	38.00%	38.00%	16.00%	38.00%	28.86%	2.00%	38.00%	16.57%	21.14%	38.00%	38.00%	8.29%	38.00%	18.57%	14.00%	38.00%	38.00%	8.29%
N = 4 M = 4	13.17%	0.90%	26.95%	7.19%	14.07%	28.44%	13.17%	15.87%	44.31%	28.44%	13.17%	29.04%	13.17%	0.90%	25.75%	7.19%	15.87%	45.81%
N = 4 M = 5	2.17%	28.92%	14.22%	1.20%	19.76%	25.54%	2.17%	2.17%	25.54%	23.86%	2.17%	16.39%	2.17%	28.92%	31.81%	19.76%	0.00%	29.40%
N = 4 M = 5	0.00%	12.89%	23.97%	0.00%	24.48%	12.89%	22.16%	0.00%	23.97%	34.28%	22.42%	22.42%	0.00%	13.14%	29.90%	11.08%	22.42%	22.42%
Moyenne	10.50%	12.57%	17.19%	9.46%	11.53%	16.72%	12.83%	11.20%	19.40%	14.67%	11.91%	16.10%	11.91%	13.71%	19.20%	14.54%	15.16%	21.05%

TABLE 5.4 – Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de la qualité de la solution obtenu

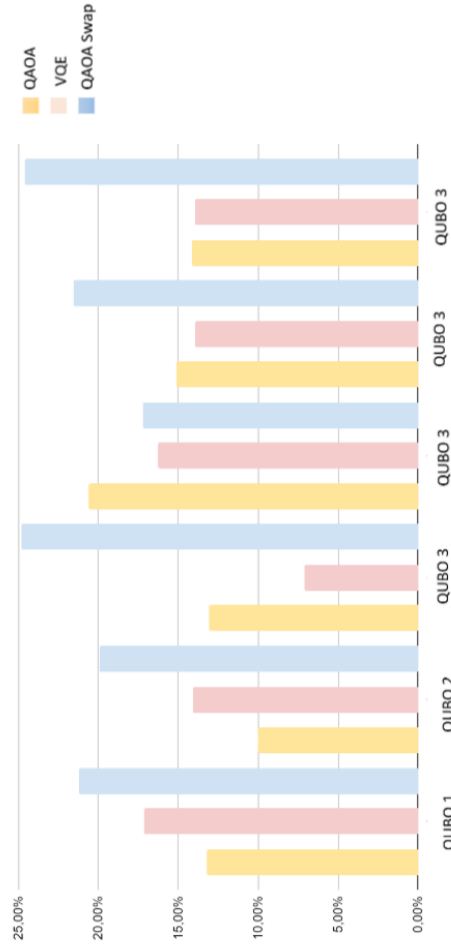


FIGURE 5.19 – Comparaison entre les méthodes implémentées QAOA, VQE et QAOA SWAP en termes de la qualité de la solution obtenu



## 5.6 Conclusion

Dans ce chapitre, nous avons présenté la partie implémentations de nos solutions. Dans un premier lieu, nous avons commencé par présenter l'approche basée sur l'algorithme de Grover. Nous avons prouvé l'efficacité de notre circuit qui était capable d'augmenter la probabilité d'apparition des solutions, les résultats obtenus ont montré aussi l'importance de bien choisir le nombre de rotations sur la qualité de la solution obtenue. Dans un deuxième lieu, nous avons étudié le coût de notre circuit de Grover, cette étude nous a permis de conclure que la partie oracle 2 qui permet de distinguer les solutions valides nécessite un nombre de qubits qui augmente exponentiellement avec le nombre de tâches. La modification de la partie initialisation du circuit de Grover par un opérateur qui génère seulement les solutions valides pourrait pallier à ce problème.

Dans un deuxième lieu, nous avons présenté la partie des méthodes variationnelles. Les résultats montrent que les modélisations QUBO 2 et QUBO 4 donnent les meilleurs résultats par rapport aux modélisations restantes. La comparaison entre les méthodes variationnelles montre que la méthode *VQE* représente le meilleur compromis entre le temps d'exécution et la qualité de la solution.

# Conclusion générale

Les travaux développés dans le cadre de ce mémoire de fin d'étude portent sur l'utilisation des algorithmes quantiques pour la résolution des problèmes d'ordonnancements de type flow-shop. Nous avons considéré le problème de permutation flow-shop avec l'objectif de la minimisation du temps de complétion de la dernière tâche sur la dernière machine (Makespan).

Nous avons mené une étude bibliographique sur les différentes méthodes quantiques existantes, nous avons aussi proposé une classification des différentes méthodes trouvées selon les types de matériel utilisé pour l'exécution de ces algorithmes ainsi qu'une synthèse des travaux existants. Nous avons aussi présenté les problèmes d'ordonnancement, leurs notations et caractéristiques ainsi qu'une synthèse des travaux qui proposent des algorithmes quantiques pour la résolution de ce type de problèmes d'optimisation. Cette étude nous a permis de conclure que le problème de flow-shop n'était pas traité auparavant avec les algorithmes quantiques variationnels et les algorithmes basés sur la recherche de Grover.

De ce fait, nous avons proposé dans ce travail deux approches différentes pour la résolution du problème de permutation du flow-shop. La première approche est basée sur la recherche du Grover. Afin d'implémenter cette approche, nous avons commencé par adapter le circuit du Grover pour trouver des solutions ayant un makespan inférieur à une borne supérieure. Dans le but de généraliser notre solution, nous avons proposé deux méthodes différentes de recherche : (1) la recherche adaptative basé sur l'algorithme de Grover (GAS) et (2) la recherche binaire avec l'algorithme de Grover (BSG). Les résultats obtenu montre qu'on peut exploiter ces algorithmes pour la résolution de permutation flow shop, mais la contrainte de matériel quantique représente un grand challenge qui limite nos tests pour les grandes instances.

Le deuxième type d'approches proposées est les algorithmes quantique variationnels, dans ce cadre, nous avons proposé deux modèles mathématiques quadratique binaire sans contraintes approximatives pour notre problème. puis nous avons proposé trois versions des algorithmes variationnels. Les résultats expérimentaux ont prouvé la possibilité de l'utilisation des méthodes présentées pour résoudre le problème étudié, mais les tests étaient faits sur des instances qui sont considérées comme des instances facile pour les algorithmes classique. Nous pouvons déduire que ces méthodes ne sont pas encore au

point de dépasser les performances des algorithmes classique est ceci est lié aux limites du matériel quantique disponible aujourd'hui, en plus à cause de ces limites c'était difficile pour nous de pouvoir étudier les performances de ces méthodes pour des instances grandes comme les instances de Taillard.

Notre travail représente une première proposition pour la résolution du problème permutation flow shop avec les méthodes du Grover et les méthodes variationnels. Il serait envisageable de pouvoir généraliser ces solutions au problème du flow-shop. Une deuxième perspective est l'amélioration du circuit de Grover proposé en introduisant des modifications au niveau de la partie d'initialisation d'une manière qui nous permet de générer que des solutions faisables, ce qui implique la réduction de la taille de l'espace de recherche. Une dernière perspective est d'étudier l'impact du changement de l'optimiseur classique sur la qualité des solutions obtenues avec les méthodes variationnelles.

# Bibliographie

- Amaro, David, Rosenkranz, Matthias, Fitzpatrick, Nathan, Hirano, Koji, & Fiorentini, Mattia. 2022a. A case study of variational quantum algorithms for a job shop scheduling problem. *EPJ Quantum Technology*, **9**(1), 5.
- Amaro, David, Modica, Carlo, Rosenkranz, Matthias, Fiorentini, Mattia, Benedetti, Marcello, & Lubasch, Michael. 2022b. Filtering variational quantum algorithms for combinatorial optimization. *Quantum Science and Technology*, **7**(1), 015021.
- Ardelean, Sebastian Mihai, & Udrescu, Mihai. 2022. Graph coloring using the reduced quantum genetic algorithm. *PeerJ Computer Science*, **8**, e836.
- Bagchi, Tapan P, Gupta, Jatinder ND, & Sriskandarajah, Chelliah. 2006. A review of TSP based approaches for flowshop scheduling. *European Journal of Operational Research*, **169**(3), 816–854.
- Baritompa, William P, Bulger, David W, & Wood, Graham R. 2005. Grover’s quantum algorithm applied to global optimization. *SIAM Journal on Optimization*, **15**(4), 1170–1184.
- Barkoutsos, Panagiotis Kl, Nannicini, Giacomo, Robert, Anton, Tavernelli, Ivano, & Woerner, Stefan. 2020. Improving variational quantum optimization using CVaR. *Quantum*, **4**, 256.
- Baykov, Ayaz, & Protasov, Stanislav. GROVER BINARY SEARCH FOR DISCRETE QUANTUM OPTIMIZATION.
- Beaulieu, Daniel, & Pham, Anh. 2021. Max-cut clustering utilizing warm-start qaoa and ibm runtime. *arXiv preprint arXiv :2108.13464*.
- Bharti, Kishor, Cervera-Liarta, Alba, Kyaw, Thi Ha, Haug, Tobias, Alperin-Lea, Sumner, Anand, Abhinav, Degroote, Matthias, Heimonen, Hermanni, Kottmann, Jakob S, Menke, Tim, *et al.* 2021. Noisy intermediate-scale quantum (NISQ) algorithms. *arXiv preprint arXiv :2101.08448*.

- Blazewicz, Jacek, Lenstra, Jan Karel, & Kan, AHG Rinnooy. 1983. Scheduling subject to resource constraints : classification and complexity. *Discrete applied mathematics*, **5**(1), 11–24.
- Bonet-Monroig, Xavier, Wang, Hao, Vermetten, Diederick, Senjean, Bruno, Moussa, Charles, Bäck, Thomas, Dunjko, Vedran, & O’Brien, Thomas E. 2021. Performance comparison of optimization methods on variational quantum algorithms. *arXiv preprint arXiv :2111.13454*.
- Boyer, Michel, Brassard, Gilles, Høyer, Peter, & Tapp, Alain. 1998. Tight bounds on quantum searching. *Fortschritte der Physik : Progress of Physics*, **46**(4-5), 493–505.
- Bulger, David, Baritomp, William P, & Wood, Graham R. 2003. Implementing pure adaptive search with Grover’s quantum algorithm. *Journal of optimization theory and applications*, **116**(3), 517–529.
- Campbell, Herbert G, Dudek, Richard A, & Smith, Milton L. 1970. A heuristic algorithm for the n job, m machine sequencing problem. *Management science*, **16**(10), B–630.
- Carlier, Jacques, Chrétienne, Philippe, Erschler, Jacques, Hanen, Claire, Lopez, Pierre, Munier, Alix, Pinson, Eric, Portmann, Marie-Claude, Prins, Christian, Proust, Christian, *et al.* 1993. Les problèmes d’ordonnancement. *RAIRO-Operations Research*, **27**(1), 77–150.
- Cerezo, Marco, Arrasmith, Andrew, Babbush, Ryan, Benjamin, Simon C, Endo, Suguru, Fujii, Keisuke, McClean, Jarrod R, Mitarai, Kosuke, Yuan, Xiao, Cincio, Lukasz, *et al.* 2021. Variational quantum algorithms. *Nature Reviews Physics*, **3**(9), 625–644.
- Cook, Jeremy, Eidenbenz, Stephan, & Bärttschi, Andreas. 2020. The quantum alternating operator ansatz on maximum k-vertex cover. *Pages 83–92 of : 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE.
- Draper, Thomas G. 2000. Addition on a quantum computer. *arXiv preprint quant-ph/0008033*.
- Durr, Christoph, & Hoyer, Peter. 1996. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*.
- Egger, Daniel J, Mareček, Jakub, & Woerner, Stefan. 2021. Warm-starting quantum optimization. *Quantum*, **5**, 479.
- EL BAHLOUL, SANA. 2008. FLOW-SHOP À DEUX MACHINES AVEC DES TEMPS DE LATENCE : APPROCHE EXACTE ET HEURISTIQUE.
- Ezratty, Olivier. *Comprendre l’informatique quantique*.

- Farhi, Edward, Goldstone, Jeffrey, Gutmann, Sam, & Sipser, Michael. 2000. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*.
- Farhi, Edward, Goldstone, Jeffrey, & Gutmann, Sam. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv :1411.4028*.
- Feynman, Richard P. 1986. Quantum mechanical computers. *Foundations of physics*, **16**(6), 507–531.
- Feynman, Richard P, Hey, Tony, & Allen, Robin W. 2018. *Feynman lectures on computation*. CRC Press.
- Gilliam, Austin, Woerner, Stefan, & Gocciulea, Constantin. 2021. Grover adaptive search for constrained polynomial binary optimization. *Quantum*, **5**, 428.
- Glover, Fred, Kochenberger, Gary, Hennig, Rick, & Du, Yu. 2022. Quantum Bridge Analytics I : a tutorial on formulating and using QUBO models. *Annals of Operations Research*, 1–43.
- Goh, Siong Thye, Gopalakrishnan, Sabrish, Bo, Jianyuan, & Lau, Hoong Chuin. 2020. A hybrid framework using a QUBO solver for permutation-based combinatorial optimization. *arXiv preprint arXiv :2009.12767*.
- Gondran, Alexandre, & Gondran, Michel. 2020. Optimisation Combinatoire et les Ordinateurs Quantiques. In : *ROADEF 2020, 21ème congrès annuel de la société Française de Recherche Opérationnelle et d'Aide à la Décision*.
- Graham, Ronald Lewis, Lawler, Eugene Leighton, Lenstra, Jan Karel, & Kan, AHG Rinnooy. 1979. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Pages 287–326 of : Annals of discrete mathematics*, vol. 5. Elsevier.
- Grover, Lov K. 1996. A fast quantum mechanical algorithm for database search. *Pages 212–219 of : Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*.
- Han, Kuk-Hyun, & Kim, Jong-Hwan. 2002. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE transactions on evolutionary computation*, **6**(6), 580–593.
- Haverly, A, & López, S. 2021. Implementation of Grover’s Algorithm to Solve the Maximum Clique Problem. *Pages 441–446 of : 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE.
- Ikeda, Kazuki, Nakamura, Yuma, & Humble, Travis S. 2019. Application of quantum annealing to nurse scheduling problem. *Scientific reports*, **9**(1), 1–10.

- Isaac Chuang, Michael Nielsen. 2002. *Quantum Computation and Quantum Information*.
- Jaffali, Hamza. 2020. *Étude de l’Intrication dans les Algorithmes Quantiques : Approche Géométrique et Outils Dérivés*. Ph.D. thesis, Bourgogne Franche-Comté.
- Khumalo, Maxine T, Chieza, Hazel A, Prag, Krupa, & Woolway, Matthew. 2021. An investigation of IBM Quantum Computing device performance on Combinatorial Optimisation Problems. *arXiv preprint arXiv :2107.03638*.
- LaRose, Ryan. 2019. Overview and comparison of gate level quantum software platforms. *Quantum*, **3**, 130.
- Layeb, Abdesslem. 2010. *Utilisation des Approches d’Optimisation Combinatoire pour la Vérification des Applications Temps Réel*. Ph.D. thesis, Thèse de Doctorat, Université Mentouri de Constantine.
- Leonidas, Giannis D. 2021. *Quantum approximate optimization algorithms and applications*. Ph.D. thesis, Technical University of Crete.
- Lewis, Mark, & Glover, Fred. 2017. Quadratic unconstrained binary optimization problem preprocessing : Theory and empirical analysis. *Networks*, **70**(2), 79–97.
- Li, Yangyang, Tian, Mengzhuo, Liu, Guangyuan, Peng, Cheng, & Jiao, Licheng. 2020. Quantum optimization and quantum learning : A survey. *Ieee Access*, **8**, 23568–23593.
- Liu, Xiaoyuan, Angone, Anthony, Shaydulin, Ruslan, Safro, Ilya, Alexeev, Yuri, & Cincio, Lukasz. 2022. Layer vqe : A variational approach for combinatorial optimization on noisy quantum computers. *IEEE Transactions on Quantum Engineering*, **3**, 1–20.
- Lu, Feng, & Marinescu, Dan C. 2007. An R|| C max Quantum Scheduling Algorithm. *Quantum Information Processing*, **6**(3), 159–178.
- Lucas, Andrew. 2014. Ising formulations of many NP problems. *Frontiers in physics*, **5**.
- Moll, Nikolaj, Barkoutsos, Panagiotis, Bishop, Lev S, Chow, Jerry M, Cross, Andrew, Egger, Daniel J, Filipp, Stefan, Fuhrer, Andreas, Gambetta, Jay M, Ganzhorn, Marc, *et al.* 2018. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology*, **3**(3), 030503.
- Montanaro, Ashley. 2016. Quantum algorithms : an overview. *npj Quantum Information*, **2**(1), 1–8.
- Moore. 1975. Moore, G. E. (1975, December). Progress in digital integrated electronics. In *Electron devices meeting* (Vol. 21, pp. 11-13).

- Mukherjee, Sayan. 2022. A grover search-based algorithm for the list coloring problem. *IEEE Transactions on Quantum Engineering*, **3**, 1–8.
- Narayanan, Ajit, & Moore, Mark. 1996. Quantum-inspired genetic algorithms. *Pages 61–66 of : Proceedings of IEEE international conference on evolutionary computation*. IEEE.
- National Academies of Sciences, Engineering, & Medicine. 2019. Quantum Computing : Progress and Prospects.
- Nawaz, Muhammad, Ensore Jr, E Emory, & Ham, Inyong. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, **11**(1), 91–95.
- Panchi, Li, & Shiyong, Li. 2008. Grover quantum searching algorithm based on weighted targets. *Journal of Systems Engineering and Electronics*, **19**(2), 363–369.
- Peruzzo, Alberto, McClean, Jarrod, Shadbolt, Peter, Yung, Man-Hong, Zhou, Xiao-Qi, Love, Peter J, Aspuru-Guzik, Alán, & O'brien, Jeremy L. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, **5**(1), 1–7.
- Pires, Otto Menegasso, de Santiago, Rafael, & Marchi, Jerusa. 2021. Two stage quantum optimization for the school timetabling problem. *Pages 2347–2353 of : 2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE.
- Qiskit, IBM, & Pattanayak, Santanu. Quantum Machine Learning with Python.
- Radzihovsky, Matthew, Murphy, Joey, & Mason, S. 2019. *A QAOA solution to the traveling salesman problem using pyQuil*.
- Rieffel, Eleanor, Dominy, Jason M, Rubin, Nicholas, & Wang, Zhihui. 2020. XY-mixers : analytical and numerical results for QAOA.
- Ross, Oscar H Montiel. 2019. A review of quantum-inspired metaheuristics : going from classical computers to real quantum computers. *IEEE Access*, **8**, 814–838.
- Scherer, Antonius, Guggemos, Tobias, Grundner-Culemann, Sophia, Pomplun, Nikolas, Prüfer, Sven, & Spörl, Andreas. 2021. OnCall Operator Scheduling for Satellites with Grover's Algorithm. *Pages 17–29 of : International Conference on Computational Science*. Springer.
- Šeda, Miloš. 2007. Mathematical models of flow shop and job shop scheduling problems. *International Journal of Physical and Mathematical Sciences*, **1**(7), 307–312.
- Shimada, Daichi, Shibuya, Toshiyuki, & Shibasaki, Takayuki. 2021. A decomposition method for makespan minimization in job-shop scheduling problem using ising machine.



- Pages 307–314 of : 2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*. IEEE.
- Shor, Peter W. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, **41**(2), 303–332.
- Spall, James C. 1998. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, **34**(3), 817–823.
- Srinivasan, Karthik, Satyajit, Saipriya, Behera, Bikash K, & Panigrahi, Prasanta K. 2018. Efficient quantum algorithm for solving travelling salesman problem : An IBM quantum experience. *arXiv preprint arXiv :1805.10928*.
- Stollenwerk, Tobias, Hadfield, Stuart, & Wang, Zhihui. 2020. QAOA for Optimal Flight Gate Assignment. *Bulletin of the American Physical Society*, **65**.
- Tabi, Zsolt, El-Safty, Kareem H, Kallus, Zsófia, Hága, Péter, Kozsik, Tamás, Glos, Adam, & Zimborás, Zoltán. 2020. Quantum optimization for the graph coloring problem with space-efficient embedding. *Pages 56–62 of : 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE.
- Theis, Thomas N, & Wong, H-S Philip. 2017. The end of moore’s law : A new beginning for information technology. *Computing in Science & Engineering*, **19**(2), 41–50.
- Umeano, Chukwudubem. 2021. *Quantum Algorithms : A Review*. Ph.D. thesis, Imperial College London.
- van Dam, Wim, Eldefrawy, Karim, Genise, Nicholas, & Parham, Natalie. 2021. Quantum optimization heuristics with an application to knapsack problems. *Pages 160–170 of : 2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE.
- Venturelli, Davide, Marchand, D, & Rojo, Galo. 2016. Job shop scheduling solver based on quantum annealing. *Pages 25–34 of : Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*.
- Vert, Daniel. 2021. *Étude des performances des machines à recuit quantique pour la résolution de problèmes combinatoires*. Ph.D. thesis, Université Paris-Saclay.
- Vinod, Gayathree M, & Shaji, Anil. 2021. Solving the integer case constraint satisfiability problem using Grover’s algorithm. *arXiv preprint arXiv :2106.09976*.
- Wittrock, Robert J. 1985. Scheduling algorithms for flexible flow lines. *IBM Journal of Research and Development*, **29**(4), 401–412.

- Zahedinejad, Ehsan, & Zaribafiyani, Arman. 2017. Combinatorial optimization on gate model quantum computers : A survey. *arXiv preprint arXiv :1708.05294*.
- Zhang, Jiachen, Bianco, Giovanni Lo, & Beck, J Christopher. 2022. Solving Job-Shop Scheduling Problems with QUBO-Based Specialized Hardware.
- Zouambi. 2019. Algorithme génétique quantique pour la résolution du problème de l'ordonnancement , PFE ESI 2019.