Échantillonnage et dénombrement avec les algorithmes variationnels quantiques

par

Julien Drapeau

Mémoire présenté au département de physique
en vue de l'obtention du grade de maître ès science (M.Sc.)

FACULTÉ des SCIENCES
UNIVERSITÉ de SHERBROOKE

Sherbrooke, Québec, Canada, 24 mars 2025

Le _____

le jury a accepté le mémoire de M. Julien Drapeau dans sa version finale.

Membres du jury

Professeur Stefanos Kourtis
Directeur de recherche
Département de physique

Professeur André-Marie Tremblay
Membre interne
Département de physique

Professeur Baptiste Royer
Président rapporteur
Département de physique

« Je vis parce que les montagnes ne savent pas rire, ni les vers de terre chanter. »

- Emil Cioran

Sommaire

En exploitant à la fois les ressources classiques et quantiques, les algorithmes variationnels quantiques peuvent exécuter des tâches computationnelles complexes sur des ordinateurs quantiques bruités, sans nécessiter de correction d'erreurs. Ces algorithmes visent notamment à résoudre des problèmes d'optimisation combinatoire, se plaçant ainsi en concurrence avec des algorithmes classiques perfectionnés au fil des décennies. Une approche complémentaire consiste à se tourner vers des problèmes intrinsèquement plus complexes : les problèmes de dénombrement. Ces derniers, qui consistent à compter le nombre de solutions aux problèmes de décision, pourraient se révéler plus accessibles aux algorithmes variationnels quantiques qu'aux algorithmes classiques.

Dans le cadre de ce mémoire, un algorithme surnommé VQCount est introduit pour la résolution de problèmes de dénombrement. VQCount s'appuie sur l'équivalence entre l'échantillonnage aléatoire et le comptage approximatif pour estimer le nombre de solutions à un facteur multiplicatif près, en exploitant l'ansatz quantique à opérateurs alternants comme générateur de solutions. Cette approche nécessite un nombre d'échantillons polynomial selon la taille du problème, même lorsque ce dernier possède un nombre exponentiel de solutions, une amélioration exponentielle par rapport à des travaux précédents.

À l'aide de simulations de réseaux de tenseurs, la performance de VQCount avec des circuits de faible profondeur est étudiée sur des instances synthétiques de deux problèmes #P-difficile, positif #1-in-3SAT et positif #NAE3SAT, en employant comme générateurs de solutions l'algorithme quantique d'optimisation approximative ainsi que l'ansatz quantique à opérateurs alternants avec forçage de Grover. Un compromis entre la probabilité de succès et l'uniformité de l'échantillonnage du générateur est observé et exploité pour atteindre un gain en efficacité exponentiel par rapport à l'échantillonnage par rejet naïf. Ces résultats soulignent le potentiel et les limites des algorithmes variationnels quantiques pour le comptage approximatif.

Mots-clés : Algorithmes variationnels quantiques, Échantillonnage aléatoire, Comptage approximatif, Auto-réductibilité, Satisfaisabilité, Réseaux de tenseurs

Remerciements

Ces dernières années furent parfois exhilarantes, parfois éprouvantes, mais j'en ressors indubitablement grandi. J'ose espérer que mon parcours ne s'arrête pas là et que j'aurais l'occasion de m'épanouir davantage. Je tiens à remercier ceux et celles qui m'ont aidé à devenir la personne que je suis aujourd'hui.

À ma mère et à mes soeurs, pour votre soutien infaillible malgré les défis rencontrés. Aucun mot ne saurait suffire pour vous remercier.

À Justin, Léanne, Thibault, Thomas, Philippe et William, votre présence à mes côtés m'est infiniment précieuse. Malgré la distance, vous ne cessez d'illuminer mes journées et mes soirées. Les déjeuners du dimanche me tiennent particulièrement à coeur, et j'espère perpétuer cette tradition !

À Antoine, Benjamin, Jérémie, Jeremy, Martin, Pierre-Alexandre et tous les membres du groupe, merci d'alléger le poids des études, de tolérer mes râleries occasionnelles et d'être toujours partant pour la discussion. Vos conseils ont été d'une aide précieuse, et nos parties de rugby et d'ultimate, des échappatoires bienvenues.

À Sovannie, pour m'avoir fait ressentir l'intensité de la vie, avec ses hauts et ses bas. Merci de m'avoir fait redécouvrir la richesse des liens humains et le plaisir des échanges sincères. Sans aucun doute, notre rencontre a changé ma vision du monde pour le meilleur.

À Stefanos, pour avoir été un superviseur de recherche irréprochable tout au long de mon parcours académique. Ta patience, ta pédagogie et ton enthousiasme ont grandement contribué à ma réussite. Merci de m'avoir fait découvrir un domaine fascinant de la physique au moment où mon intérêt pour la discipline s'estompait. Au plaisir de continuer cette collaboration !

Enfin, merci à tous ceux et celles qui, d'une manière ou d'une autre, ont marqué cette surprenante période de ma vie.

Table des matières

Liste des figures

Introduction

En exploitant les principes du parallélisme, de la superposition et de l'intrication, les algorithmes quantiques remettent en question les limites classiques du calcul, promettant des accélérations exponentielles pour certains types de problèmes. L'algorithme de Shor, par exemple, permet de factoriser des entiers naturels en offrant une accélération superpolynomiale par rapport aux meilleurs algorithmes classiques connus [1]. Bien que le calcul quantique soit prometteur, son utilité pratique reste floue pour diverses raisons. Les algorithmes quantiques ne s'appliquent en ce moment qu'à une classe de problèmes restreints, et leurs applications restent limitées. En outre, les ordinateurs quantiques actuels, appelés ordinateurs quantiques à échelle intermédiaire bruités, sont affligés par différentes complications rendant tout calcul excessivement difficile : un nombre restreint de qubits, une connectivité limitée et la présence d'erreurs entravant la taille des circuits.

Pour contourner ces difficultés, les algorithmes variationnels quantiques (VQA) furent conçus pour exploiter les mécanismes du calcul quantique, tout en tirant profit de la puissance du calcul classique [2]. Ces algorithmes visent à résoudre des problèmes complexes en utilisant l'état préparé par un circuit quantique paramétré, dont les paramètres sont ajustés par un optimiseur classique pour minimiser la fonction de coût du problème. Étant basée sur l'optimisation de paramètres, cette stratégie permet de considérer toutes les difficultés précédentes d'un seul coup, tout en limitant la taille des circuits quantiques. Au cours des dernières années, de nombreux travaux ont tenté de perfectionner les VQA en développant par exemple des techniques permettant une meilleure exploitation de la structure des problèmes. Toutefois, il reste encore à déterminer si ces algorithmes présentent un réel avantage par rapport aux algorithmes classiques.

Les VQA sont principalement utilisés pour résoudre des problèmes d'optimisation combinatoire, tels que le problème de coupe maximale et le problème des ensembles indépendants. Comme des solveurs classiques pour ce type de problème ont été développés au fil des décennies, il est difficile de démontrer la supériorité des VQA. Le projet de ce mémoire propose une approche différente pour aborder cette question. Plutôt que de rivaliser avec des algorithmes classiques hautement performants, les VQA sont appliqués à la résolution de problèmes vastement plus complexes : les problèmes de dénombrement, ou plus simplement les problèmes de comptage. Ceux-ci sont difficiles même pour les algorithmes classiques, mais potentiellement moins pour les algorithmes quantiques. Au lieu de chercher une solution quasi optimale, les problèmes de dénombrement s'intéressent plutôt à déterminer le nombre de solutions respectant les contraintes du problème. Ces problèmes, qui font partie de la classe de complexité #P, se situent au-dessus des problèmes typiques d'optimisation dans la hiérarchie de la complexité computationnelle.

Pour résoudre les problèmes de dénombrement, l'équivalence entre le comptage approximatif et l'échantillonnage quasi uniforme, établie par Jerrum, Valiant et Vazirani [3], est utilisée. Cette correspondance donne lieu à un algorithme randomisé de comptage approximatif, surnommé algorithme de JVV, permettant l'estimation du nombre de solutions d'un problème de comptage auto-réductible à un facteur multiplicatif près *si* les solutions peuvent être échantillonnées suffisamment uniformément. Cet algorithme exploite la propriété d'auto-réductibilité de certains problèmes, c'est-à-dire la capacité à utiliser la structure inhérente au problème pour sa résolution.

L'algorithme VQCount, introduit dans ce mémoire, fait le pont entre les VQA et les problèmes de comptage. Cet algorithme montre qu'il est possible d'utiliser les VQA pour construire un solveur approximatif aux problèmes #P. Pour ce faire, VQCount prépare une distribution d'états contenant une superposition de solutions au problème de comptage avec une haute probabilité en utilisant un VQA nommé ansatz quantique à opérateurs alternants (QAOA) [4]. Après avoir modifié le circuit quantique de QAOA pour tirer avantage de la propriété d'auto-réductibilité, VQCount emploie l'algorithme de comptage approximatif pour estimer le nombre de solutions à partir de la distribution préparée. Si QAOA échantillonne des solutions suffisamment près de l'uniformité avec un taux de succès fini, alors seulement

un nombre polynomial d'échantillons est nécessaire pour estimer le nombre de solutions avec l'algorithme de JVV.

En employant les réseaux de tenseurs pour la simulation de circuits quantiques de faible profondeur, ce projet évalue la performance de VQCount sur des instances synthétiques de deux problèmes #P-difficile : positif #1-in-3SAT et positif #NAE3SAT. Les solutions sont générées à l'aide de l'algorithme quantique d'optimisation approximative (QAOA) [5] et de l'ansatz quantique à opérateurs alternants avec forçage de Grover (GM-QAOA) [6]. Cette analyse met en évidence un équilibre entre la probabilité de succès et l'uniformité de l'échantillonnage, qui permet d'obtenir un gain exponentiel en efficacité par rapport à une approche naïve par rejet. Ces résultats révèlent à la fois les atouts et les limites des algorithmes variationnels quantiques pour le comptage approximatif.

Le chapitre 1 explore en détail les problèmes de dénombrement avec le langage de la théorie de la complexité. L'algorithme de JVV et ses concepts sous-jacents, c'est-à-dire l'auto-réductibilité, l'échantillonnage aléatoire et le comptage approximatif, sont présentés au chapitre 2. Les algorithmes variationnels quantiques sont présentés de manière générale au chapitre 3 avec un accent sur QAOA et GM-QAOA. L'algorithme VQCount, fruit de ce travail, est détaillé au chapitre 4. La performance de VQCount est finalement caractérisée au chapitre 5. L'annexe A dérive une expression fermée de l'état préparé par un circuit GM-QAOA, alors que l'annexe B introduit les méthodes de réseaux de tenseurs utilisées pour la simulation de circuit quantique de l'algorithme VQCount.

Le travail effectué au cours de ce projet a mené à la publication d'un manuscrit sur arXiv : https://arxiv.org/abs/2503.07720. Ce mémoire prend ainsi son inspiration de cet écrit.

Chapitre 1

Complexité du dénombrement

Qu'est-ce qu'un problème de dénombrement ? Simplement, un tel problème consiste à compter le nombre de solutions respectant un ensemble de contraintes. Les problèmes de dénombrement, ou de comptage, font partie des problèmes computationnels difficiles, c'est-à-dire qu'il est peu probable qu'ils soient résolubles efficacement. Les problèmes de comptage surgissent dans de nombreuses disciplines, avec des applications en raisonnement probabiliste [7-9], en fiabilité des réseaux [10, 11], en mécanique statistique [12] et en intelligence artificielle [13]. Contrairement à certains problèmes étudiés dans le domaine du calcul quantique, comme l'échantillonnage de bosons [14] ou l'échantillonnage de circuits aléatoires [15], une solution à ces problèmes peut avoir un impact concret, justifiant alors l'étude de tels problèmes.

Ce chapitre décrit le cadre théorique entourant la classe des problèmes de dénombrement en détail. Pour ce faire, celle-ci est définie à l'aide du langage de la théorie de la complexité à la section 1.1. Le problème de comptage prototypique, le problème de satisfaisabilité, sert d'exemple et est décrit à la section 1.2. Celui-ci est d'une grande importance pour ce projet et est ainsi référencé tout au long de ce mémoire. La section 1.3 explique l'importance des méthodes approximatives alors que la section 1.4 décrit les solveurs modernes pour les problèmes de comptage. Finalement, le comportement des instances aléatoires de ces problèmes est présenté à la section 1.5, où des transitions de phase font étonnamment leurs apparitions.

Il est souvent favorable de décrire la théorie de la complexité à l'aide du concept

de *machine de Turing*. Cependant, ce mémoire suppose que la thèse de Church-Turing soit vraie, et donc que n'importe quel modèle de calcul puisse être utilisé de manière équivalente, pour faire abstraction de ce langage et simplifier les idées présentées.

1.1 Classes de complexité

La théorie de la complexité s'intéresse à la classification des problèmes algorithmiques en *classes de complexité*, c'est-à-dire en ensembles de problèmes de même complexité. Classifier un problème permet de caractériser les ressources nécessaires pour sa résolution par un algorithme. Les problèmes d'une même classe ont une difficulté intrinsèque similaire, ce qui facilite le choix d'un algorithme et de ressources appropriées. Savoir qu'un problème n'est pas réalistement résoluble, ou plus précisément intraitable, limite les attentes. Sachant ceci, la recherche dans cette direction s'avère grandement utile. La théorie de la complexité cherche aussi à comparer les problèmes de différentes complexités pour comprendre l'espace des problèmes en plus grande profondeur. Comparer des problèmes faciles avec des problèmes difficiles peut par exemple aider à comprendre ce qui rend un problème difficile et donc à trouver des algorithmes résolvant efficacement les problèmes plus complexes. Des liens, nommés réductions, sont définis entre différents problèmes pour comparer leurs complexités. Un algorithme efficace pour un problème peut en effet être efficace pour un différent problème s'il existe une réduction entre ceux-ci. Les classes de complexité, de manière similaire au modèle de la machine de Turing, tentent de définir de manière abstraite la difficulté d'un problème. Quel que soit le matériel informatique à notre disposition, un problème d'une classe donnée ne doit pas changer de classe ; un problème simple doit rester simple dans tous les cas.

Comment est-il possible de déterminer la complexité d'un problème ? Pour ce faire, les classes de complexité se basent sur les ressources indispensables à la résolution du problème : le temps et la mémoire. Afin de trouver la solution à un problème, un programme doit effectuer un certain nombre d'opérations, limité dans le temps par le matériel informatique. On parle alors de *complexité en temps*. Afin de produire un résultat final, le programme doit aussi garder en mémoire les résultats intermédiaires. Ceux-ci doivent être sauvegardés dans le matériel informatique afin

d'être réutilisés ultérieurement. Comme la quantité d'information conservée est aussi un facteur limitant pour le matériel informatique, on parle donc de *complexité en espace*.

La complexité en temps et en espace d'un problème est définie selon la taille de celui-ci. Afin de capturer cette dépendance, une loi d'échelle est utilisée pour encapsuler la difficulté d'un problème en fonction de sa taille. Le temps et la mémoire quantifient bien les ressources nécessaires des algorithmes. Par contre, ceux-ci dépendent du matériel informatique utilisé. Il est attendu qu'un ordinateur moderne soit bien plus performant qu'une des premières machines analogues. Comment retirer cette dépendance de la notion de complexité ? Pour ce faire, on fait appel à la notation asymptotique, communément appelée la *notation* $\mathcal{O}$. La notation asymptotique caractérise la vitesse de croissance d'une fonction en ne considérant que son comportement global à l'infini. Les coefficients ainsi que les termes asymptotiquement inférieurs ne sont pas considérés. Par exemple, pour un problème de taille n, la résolution de celui-ci pourrait demander un temps exponentiel $\mathcal{O}(2^n)$ et une mémoire polynomiale $\mathcal{O}(n)$. Remarquons qu'il n'y a aucune dépendance au matériel informatique ; deux ordinateurs différents doivent effectuer le même nombre d'opérations et sauvegarder la même quantité d'information. Un de ces ordinateurs pourrait toutefois résoudre le problème plus rapidement si celui-ci peut effectuer un plus grand nombre d'opérations par seconde ou accéder plus rapidement à sa mémoire. L'attrait des classes de complexité vient donc en partie de cette abstraction du matériel informatique.

La quantification de ces ressources permet la séparation de plusieurs problèmes : il est en effet souhaitable de séparer les algorithmes efficaces de ceux qui ne le sont pas. Commençons par définir deux classes de complexité particulièrement importantes : P et NP. Pour ce faire, un certain type de problème doit d'abord être défini. Un *problème de décision* regroupe simplement tous les problèmes pouvant se répondre par oui ou non. Tout problème de décision A peut être représenté par une fonction $A(x) \in \{0, 1\}$, où x représente une instance du problème. Les problèmes de décision se manifestent fréquemment, tant en informatique qu'en physique. Ceux-ci se présentent sous diverses formes : est-ce qu'un nombre x est premier ? La configuration x représente-t-elle un état fondamental du système donné ? Est-ce qu'il existe un chemin x parcourant une seule fois toutes les villes d'une région en

parcourant au maximum une distance d ?

Quand peut-on dire qu'un problème de décision est résoluble efficacement ? La classe de complexité P, pour « temps polynomial », tente de répondre à cette question. Informellement, un problème de la classe P est un problème de décision qui peut être résolu en temps polynomial. Un problème est donc considéré comme efficacement résoluble ou *traitable*, s'il appartient à la classe P.

Définition 1.1 – Classe de complexité P

Une fonction A fait partie de la classe de complexité P si et seulement si elle prend la forme

$$A(x) = \exists y$$

et elle est calculable en temps polynomial, c'est-à-dire en temps $O(n^c)$ pour une taille $n = |x|$ et une constante c, où $|y| = \text{poly}(|x|)$.

Soit, par exemple, le problème de décision du test de primalité A. Ce problème cherche à déterminer si un entier naturel x est premier ou composé. Ce problème peut être résolu, c'est-à-dire qu'il est possible de calculer $A(x) = \exists y$, en temps polynomial $\tilde{O}(\log(n)^{12})$ [16], où la notation $\tilde{O}$ signifie que les termes polylogarithmiques sont aussi cachés.

La relation entre un calcul en temps polynomial et un calcul efficace semble évidente de prime abord. La thèse de Cobham–Edmonds indique en effet qu'un problème algorithmique peut être résolu efficacement s'il est résoluble en temps polynomial [17, 18]. Cependant, certains problèmes ne possèdent pas de solutions trouvables efficacement en pratique. Par exemple, un problème peut appartenir à la classe P, mais être doté d'un grand coefficient limitant le calcul. Cela n'étant pas le cas pour la majorité des problèmes, cette supposition s'avère malgré tout une bonne règle empirique.

Une deuxième classe particulièrement importante en théorie de la complexité est la classe NP, pour « temps polynomial non déterministe ». Celle-ci regroupe les problèmes de décision dont les solutions sont vérifiables en temps polynomial. Généralement, ces problèmes sont formulés sous la forme suivante : existe-t-il une solution, vérifiable en temps polynomial, au problème donné ? Bien que les solutions

soient vérifiables efficacement, déterminer l'existence d'une telle solution n'est pas nécessairement possible en temps polynomial. Une métaphore souvent utilisée pour la description d'un problème NP est celle d'une aiguille dans une botte de foin. Trouver cette aiguille parmi la quantité énorme de brins de foin est un défi de taille. Par contre, une fois l'aiguille trouvée, il n'y a aucun doute qu'il s'agit bien d'une aiguille.

Définition 1.2 – Classe de complexité NP

Une fonction A fait partie de la classe de complexité NP si et seulement si elle prend la forme

$$A(x) = \exists y \mid B(x, y)$$

pour une fonction $B \in \mathsf{P}$, où $|y| = \text{poly}(|x|)$.

On appelle B le vérificateur du problème de décision A et y le certificat ou le témoin pour l'entrée x. Un exemple de problème NP est le problème du commis voyageur. Soit un graphe x représentant les villes d'une région particulière. Est-ce que le commis voyageur peut visiter toutes les villes exactement une fois et revenir à son point de départ en parcourant une distance inférieure à d ? Dans ce cas, la fonction A détermine s'il existe un tel chemin alors que la fonction B détermine si le chemin y est de taille inférieure à d.

La classe NP est également définie comme la classe de problèmes que peut résoudre une machine de Turing non déterministe en temps polynomial, d'où son nom. Cette définition est toutefois équivalente à la définition précédente, c'est-à-dire la classe de problèmes vérifiable par une machine de Turing déterministe en temps polynomial [19]. Notons que la classe P est contenue dans la classe NP. En effet, comme un problème de P est résoluble en temps polynomial, alors nécessairement une solution à ce problème peut aussi être vérifiée en temps polynomial. Cependant, une question, faisant partie des problèmes du prix du millénaire [20], demeure ouverte : est-ce que $\mathsf{P} = \mathsf{NP}$? La conjecture la plus largement répandue répond à cette question par la négative. L'hypothèse de temps exponentiel [21] suggère par exemple que certains problèmes de NP sont résolubles en temps exponentiel, un résultat significatif sur la difficulté de ces problèmes. En conséquence, un problème est *intraitable* s'il n'est pas résoluble en temps polynomial, par opposition à un

problème traitable.

La classe de complexité d'intérêt dans le cadre de ce mémoire est la classe #P introduite par Valiant [22]. Cette classe, définie par extension à la classe NP, cherche non seulement à déterminer si un problème de décision possède une solution, mais aussi à spécifier le nombre de solutions à ce problème. Ainsi, un *problème de dénombrement* ou un *problème de comptage* consiste à trouver le nombre de solutions d'un problème de décision. Un problème de comptage est donc défini par rapport à un problème de décision. Prenons par exemple le problème du commis voyageur. Le problème de décision généralement défini cherche un trajet de taille inférieure à une certaine distance. Le problème de comptage adjoint exige alors le nombre de trajets possibles de taille inférieure à la distance donnée.

Définition 1.3 – Classe de complexité #P

Une fonction A fait partie de la classe de complexité #P si et seulement si elle prend la forme

$$A(x) = |\{ y \mid B(x,y) \}|$$

pour une fonction $B \in \mathsf{P}$, où $|y| = \text{poly}(|x|)$ pour toutes les valeurs y prises par $B(x,y)$.

Par définition, un problème de la classe #P est au moins aussi difficile qu'un problème de la classe NP. Connaître le nombre de solutions à un problème de décision indique effectivement s'il existe au moins une solution à un problème de décision. Cependant, les problèmes de comptage intéressants possèdent fréquemment un nombre exponentiel de solutions, impliquant une plus grande complexité inhérente.

Comme que les problèmes de la classe P appartiennent aussi à la classe NP, et que les problèmes de cette dernière appartiennent aussi à la classe #P, il est nécessaire de définir un concept supplémentaire spécifiant la difficulté d'un problème au sein d'une classe. Intuitivement, un problème est *difficile* pour une classe de complexité s'il est au moins aussi difficile que tous les problèmes de la classe. Plus formellement, un problème difficile pour une classe donnée signifie qu'il existe une réduction en temps polynomiale de tous les problèmes de la classe au problème difficile. Une *réduction en temps polynomial* d'un problème à un autre signifie qu'il est possible d'utiliser le second problème pour résoudre le premier au coût d'un facteur polynomial. De plus,

un problème est *complet* s'il est difficile et aussi membre de la classe. La figure 1.1 éclaircit ces concepts. Les problèmes complets d'une classe représentent alors les problèmes les plus difficiles de celle-ci. Par exemple, le problème du commis voyage est dit NP-complet, car il fait partie des problèmes considérés comme difficiles de la classe NP. Il est toutefois cru que celui-ci ne fasse pas partie de la classe P selon la conjecture $\mathsf{P} \neq \mathsf{NP}$.

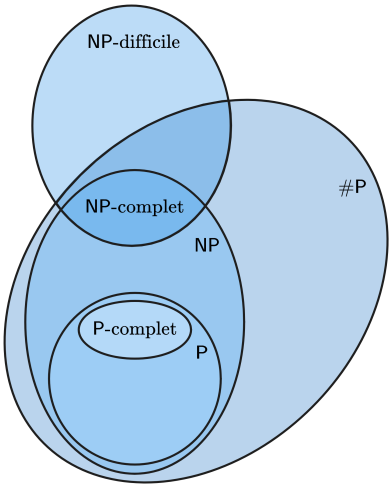

FIGURE 1.1 – Schéma de différentes classes de complexité. Ce schéma suppose la conjecture $\mathsf{P} \neq \mathsf{NP}$. Pour plus de rigueur, la classe #P devrait être remplacée par la classe $\mathsf{P}^{\#\mathsf{P}}$.

L'importance des problèmes #P au sein de la théorie de la complexité s'observe entre autres par le théorème de Toda [23]. Ce résultat marquant montre que tous les problèmes de la hiérarchie polynomiale PH peuvent être résolus en temps polynomial avec un oracle résolvant instantanément les problèmes #P. La hiérarchie polynomiale généralise les classes de complexité P, NP et co-NP en capturant les classes de problèmes exprimés par une alternance de quantificateurs d'existence ($\exists$) ou d'universalité ($\forall$) (voir la définition 1.2 pour un exemple contenant un seul quantificateur d'existence). Ainsi, la hiérarchie PH est comprise dans la classe de complexité $\mathsf{P}^{\#\mathsf{P}}$, c'est-à-dire les problèmes résolubles avec un oracle #P. Comme la hiérarchie polynomiale contient de nombreux problèmes importants, ce théorème suggère l'incroyable puissance des problèmes de comptage.

Théorème 1.1 – Théorème de Toda

La hiérarchie polynomiale PH est contenue dans $\mathsf{P}^{\#\mathsf{P}}$.

L'ordinateur quantique bouleverse le domaine de la complexité classique. Les problèmes autrefois intraitables deviennent potentiellement résolubles efficacement à l'aide d'algorithmes quantiques. Deux nouvelles classes de complexité font alors leurs apparitions pour décrire les problèmes résolubles à l'aide de matériel informatique quantique. La classe BQP généralise la classe BPP pour les ordinateurs quantiques, où la classe BPP, pour « temps polynomial probabiliste à erreur bornée », est l'ensemble des problèmes résolubles avec une probabilité d'erreur inférieure à $1/3$. De plus, la classe de complexité QMA, pour « Merlin Arthur Quantique » se définit par rapport à la classe BQP de manière analogue à la classe NP pour la classe P. La théorie de la complexité quantique étudie les classes de complexité quantiques dans l'objectif de déterminer les problèmes où les algorithmes quantiques présentent un avantage par rapport aux algorithmes classiques. Ce mémoire avance la recherche dans cette direction en étudiant la différence en performance des algorithmes quantiques et des algorithmes classiques.

1.2 Satisfaisabilité booléenne

Le problème de *satisfaisabilité booléenne* ou problème SAT est particulièrement important dans la théorie de la complexité. Montré comme NP-complet par le théorème de Cook-Levin [24, 25], il fut à la base de la définition de NP-complétude et du problème $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$. Celui-ci est aussi couramment utilisé dans la preuve de réductions de problèmes au sein de la classe de complexité NP. Le problème SAT a une multitude d'applications, comme le diagnostic des fautes d'un circuit logique ou la planification en intelligence artificielle [26], en partie grâce à la facilité de formuler ces applications à l'aide de formules propositionnelles.

Une *formule propositionnelle*, ou une expression booléenne, est un ensemble de variables booléennes $x_i \in \{\texttt{FAUX}, \texttt{VRAI}\}$ reliées par des opérateurs booléens de conjonctions (OU, $\vee$), de disjonctions (ET, $\wedge$) ainsi que de négation (NON, $\neg$). Un *littéral* désigne dans ce contexte une variable booléenne ou sa négation. Par exemple,

l'expression $(x_1 \wedge x_2) \vee \neg x_3$ est une formule booléenne composée des variables x_1, x_2 et x_3, ainsi que des littéraux x_1, x_2 et $\neg x_3$. Notons que l'équivalence `FAUX` $\leftrightarrow 0$ et `VRAI` $\leftrightarrow 1$ est utilisée dans ce mémoire par commodité.

Un problème SAT se décrit par une formule propositionnelle. Résoudre le problème consiste à déterminer s'il existe une combinaison de variables qui rend la formule logiquement vraie, c'est-à-dire que l'évaluation de celle-ci donne 1. Une telle formule est alors dite satisfaisable.

Définition 1.4 – Problème SAT

Soit une constante $n \geq 1$ et une formule propositionnelle $\varphi(x_1, x_2, \ldots, x_n)$ où $x_i \in \{0, 1\}$. Existe-t-il une assignation des variables $x_1, x_2, \ldots, x_n$ pour laquelle l'expression φ soit satisfaisable, c'est-à-dire que $\varphi(x_1, x_2, \ldots, x_n) = 1$?

Dans l'étude du problème SAT, les formules propositionnelles sont souvent exprimées en *forme normale conjonctive* (« Conjunctive Normal Form ») (CNF). On parle alors de formules CNF. Celles-ci consistent en une conjonction d'une ou de plusieurs *clauses*, où une clause est une disjonction d'un ou plusieurs littéraux. Cela implique que toute clause doit contenir au moins un littéral évaluant à 1 pour que la formule soit satisfaisable. Toute formule propositionnelle peut être réécrite en forme normale conjonctive en utilisant les lois de l'algèbre booléenne.

Exemple 1.1 – Problème SAT

La formule CNF

$$\varphi(x_1, x_2, x_3) = (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$$

est satisfaisable car $\varphi(1, 0, 0) = (1 \vee 0) \wedge (\neg 1 \vee 0 \vee \neg 0) = 1$. Au contraire, la formule CNF

$$\varphi(x_1) = (x_1) \wedge (\neg x_1)$$

n'est pas satisfaisable car $\varphi(x_1) = 0$ peu importe le choix de x_1.

Le problème kSAT constitue un cas spécial du problème SAT où le nombre de littéraux appartenant à chaque clause d'une formule CNF est restreint à k littéraux au maximum. Notons que le problème kSAT est trivial pour $k = 1$, résoluble en

temps linéaire pour $k = 2$ [27], et NP-complet pour $k \geq 3$ [28]. Surprenamment, les problèmes de comptage correspondants, #2SAT et #3SAT, appartiennent tous deux à la classe #P-complet [10]. Ainsi, compter le nombre de solutions à un problème NP-complet peut être difficile même s'il est possible de trouver une solution efficacement. Intuitivement, cette difficulté provient du nombre exponentiel de solutions que possèdent les problèmes #2SAT et #3SAT.

Dans ce mémoire, deux variantes de 3SAT sont portées à l'étude : le problème Pas-Tous-Égaux 3SAT (« Not-All-Equal 3-Satisfiability ») (NAE3SAT) et le problème 1-dans-3 3SAT (« One-in-Three 3-Sastisfiability ») (1-in-3SAT). Comme le problème SAT, ces problèmes appartiennent aussi à la classe de complexité NP-complet. Les versions monotones de ces problèmes, où la négation de variables n'est pas permise, sont étonnamment aussi NP-complet par le théorème de dichotomie de Schaefer [29]. Ces problèmes de décision peuvent être associés aux problèmes de comptage #NAE3SAT et #1-in-3SAT de la classe de complexité #P.

Définition 1.5 – Problème NAE3SAT

Soit une formule CNF $\varphi(x_1, x_2, \ldots, x_n)$ pour laquelle chaque clause C contient au maximum 3 littéraux. Existe-t-il une assignation des variables $x_1, x_2, \ldots, x_n$ telle que φ soit satisfaisable tout en s'assurant que tous les littéraux de chaque clause C ne soient pas logiquement égaux ?

Notons qu'un problème NAE3SAT peut être décrit par un problème 3SAT. Soit une formule CNF f exprimant un problème 3SAT. Une formule CNF g, représentant le problème NAE3SAT associé à f, est construite en transformant chaque clause $(x \vee y \vee z)$ de f en $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$. Cette nouvelle contrainte renforce alors la condition supplémentaire, c'est-à-dire que les littéraux de chaque clause ne peuvent être tous égaux. Cette relation illustre bien une symétrie cachée derrière le problème NAE3SAT ; chacune des variables possède en moyenne la même probabilité d'être vraie ou fausse.

Définition 1.6 – Problème 1-in-3SAT

Soit une formule CNF $\varphi(x_1, x_2, \ldots, x_n)$ pour laquelle chaque clause C contient au maximum 3 littéraux. Existe-t-il une assignation des variables $x_1, x_2, \ldots, x_n$ telle que φ soit satisfaisable tout en s'assurant qu'exactement un littéral de chaque clause C soit logiquement vrai ?

De la même façon que pour le problème NAE3SAT, le problème 1-in-3SAT se transforme en un problème 3SAT en transformant chaque clause $(x \vee y \vee z)$ en $(x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$, de manière à encoder la contrainte additionnelle du problème 1-in-3SAT.

1.3 Intraitabilité, approximation et optimisation

Les problèmes algorithmiques des classes de complexité NP-difficile et #P-difficile sont considérés comme intraitables en raison de leur complexité intrinsèque. En effet, il est improbable qu'un algorithme puisse résoudre exactement ces problèmes en temps polynomial. Par exemple, bien que certaines instances du problème SAT contenant plus d'un million de variables soient résolubles efficacement, d'autres instances de moins d'un millier de variables ne peuvent être résolues par les solveurs de pointe [30]. Cette difficulté s'accentue pour le problème #SAT, où même une centaine de variables peut s'avérer trop complexe. L'intraitabilité de tels problèmes mène alors à un paradigme différent ; sachant qu'il est irréaliste de résoudre certains problèmes exactement, est-ce qu'il est possible de trouver une solution approximative de manière efficace ? L'exactitude des solutions est alors sacrifiée pour une performance accrue.

Les problèmes de décision ne possédant que deux solutions, oui ou non, il n'est donc pas possible de fournir une solution approximative. Un *problème d'optimisation*, défini en conjonction à un problème de décision, est alors une notion plus adéquate pour incorporer la notion d'approximation. Ces problèmes ne cherchent plus à trouver la solution optimale, mais plutôt à obtenir une réponse suffisamment près de la valeur optimale. Un exemple de problème d'optimisation, particulièrement étudié

dans le domaine de l'optimisation quantique en raison de sa simplicité d'implémentation sur du matériel informatique quantique, est le problème de coupe maximum (« Maximum Cut ») (Max-Cut). Ce problème cherche une coupe séparant les sommets d'un graphe en deux ensembles complémentaires, tel que le nombre d'arêtes séparant les deux ensemble soit maximal. Le problème Max-Cut étant NP-complet, il est difficile d'obtenir une coupe optimale, mais une coupe sous-optimale peut possiblement être trouvée efficacement.

Un problème d'optimisation Π est constitué d'un ensemble d'instances valides I, où chaque instance $x \in I$ possède un ensemble de solutions faisables $S(x)$. Une *fonction objectif* obj, aussi nommée fonction de coût ou de perte, quantifie la qualité d'une solution approximative y à l'instance x en lui assignant un nombre rationnel non négatif. Par conséquent, résoudre approximativement un problème d'optimisation consiste à minimiser ou maximiser la fonction de coût. Pour simplifier la notation, les définitions du reste de la section ne concernent que les problèmes de minimisation, comme la généralisation au problème de maximisation est simple. Une mesure de succès fréquemment utilisée autant classiquement que quantiquement est le *rapport d'approximation*

$$\alpha(x, y) = \frac{\text{obj}(x, y)}{\text{OBJ}(x)}, \tag{1.1}$$

où $\text{OBJ}(x) = \min_y \text{obj}(x, y)$. Ce type de problème est formalisé sous le nom de problème d'optimisation NP, mais une définition détaillée est évitée ici par simplicité. Bien que les algorithmes d'approximation heuristiques offrent fréquemment de bons résultats, ceux-ci ne possèdent aucune garantie quant à la qualité de ces solutions.

Pour pallier ce problème, des algorithmes approximatifs avec garanties peuvent être définis. Ceux-ci se basent sur deux paramètres : la tolérance ε et la confiance δ. La tolérance indique l'erreur multiplicative maximale de l'approximation et la confiance indique la probabilité de succès de l'algorithme. Ce couple de paramètres est souvent utilisé pour décrire les algorithmes approximatifs. Un *algorithme d'approximation de tolérance* ε utilise cette notion pour garantir une solution quasi optimale à une erreur multiplicative près de la solution optimale [31]. Pour un problème de minimisation Π, un tel algorithme produit une solution y pour toutes instances $x \in I$ telle que $\text{obj}(x, y) \leq \varepsilon(|x|) \cdot \text{OBJ}(x)$ où $|x|$ est la taille de l'instance et $\varepsilon \geq 1$. Cette définition peut être détendue en permettant à l'algorithme de produire une telle solution avec

une certaine probabilité. Cette variante, l'*algorithme d'approximation randomisé de tolérance* ε, introduit des concepts pertinents pour le chapitre 2.

Théorème 1.2 – Algorithme d'approximation randomisé

Un algorithme d'approximation randomisé de tolérance ε pour un problème de minimisation Π est un algorithme aléatoire prenant en entrée une instance $x \in I$ et une tolérance ε et qui produit une solution $y \in S(x)$ tel que

$$\Pr[\text{obj}(x, y) \leq \varepsilon(|x|) \cdot \text{OBJ}(x)] \geq \frac{1}{2}$$

en temps polynomial selon la taille $|x|$ de l'instance.

En relaxant la condition d'exactitude, il est attendu qu'un gain soit possible en matière d'efficacité. Cependant, cela n'est pas toujours aussi clair. Il existe en effet certains problèmes où trouver une solution approximative en haut d'un certain rapport d'approximation demeure intraitable, comme le problème de couverture par ensembles [32]. Les algorithmes d'approximation s'appliquent aussi aux problèmes de comptage tel que présenté à la section 2.3.

1.4 Comptage de modèles

Sachant désormais que le comptage est un problème difficile, comment résoudre celui-ci ? Le dénombrement des solutions d'une formule propositionnelle est spécifiquement connu dans la littérature sous le nom de *comptage de modèles* ou dénombrement de modèles. De nombreuses méthodes, autant exactes qu'approximatives, ont été développées pour sa résolution [33].

Les méthodes exactes attaquent généralement le problème en explorant exhaustivement l'espace des solutions possibles, de manière similaire à l'algorithme de Davis-Putnam-Logemann-Loveland (DPLL) pour les problèmes SAT [34]. Les algorithmes de recherche locale pour SAT, tel l'algorithme WalkSAT, peuvent aussi être étendus pour résoudre #SAT. Toutefois, les solveurs les plus performants actuellement sont basés sur les réseaux de tenseurs [35-37], un objet mathématique provenant du domaine de la matière condensée. Ces réseaux sont d'ailleurs utilisés

dans ce travail pour la simulation de circuits quantiques et font donc l'objet de l'annexe B. Du côté théorique, des algorithmes bornent, dans le pire des cas, #2SAT en temps $O(1.2377^n)$ [38] et #3SAT en temps $O(1.6737^n)$ [39] pour un problème de n variables. Bien que différents développements accentuent la taille des systèmes résolubles exactement, les solveurs exacts ont de la difficulté à trouver l'ensemble de solutions de l'espace de recherche.

Les méthodes approximatives allègent ce problème à l'aide d'heuristiques fournissant des estimations avec ou sans garanties. Plusieurs applications du comptage n'exigent pas un résultat parfaitement exact ; distinguer la différence entre 10^{30} et $10^{30} + 1$ solutions n'est pas toujours significatif. L'algorithme de Stockmeyer, qui approxime le nombre de solutions à un facteur deux avec un nombre polynomial d'appels à un oracle NP en s'appuyant sur les fonctions de hachage, fut un pas majeur pour le comptage approximatif [40]. Plusieurs modifications à cet algorithme ont propulsé le domaine vers des performances accrues de sorte que la majorité des méthodes modernes se basent effectivement sur les fonctions de hachage. Cependant, une méthode alternative due à Jerrum, Valiant et Vazirani utilise plutôt la relation entre l'échantillonnage aléatoire de solutions et le comptage approximatif pour résoudre le problème. Malgré l'originalité de cette idée, la recherche dans cette direction s'est estompée en raison de la difficulté d'échantillonner des solutions avec les conditions requises. Ce travail relance cette possibilité en pourvoyant une nouvelle façon d'échantillonner avec les algorithmes variationnels quantiques. Une discussion approfondie de ce type d'algorithme est présentée au chapitre 2.

Les avancées en physique, tels les réseaux de tenseurs, présentent des avantages pour le comptage de modèle, mais l'inverse est aussi vrai. Le problème du comptage est intimement relié au domaine de la mécanique statistique. En effet, déterminer la fonction de partition d'un système à température nulle est en réalité équivalent à un problème de comptage [41]. Considérons la fonction de partition Z pour un Hamiltonien H où l'énergie fondamentale est fixée à $E_0 = 0$,

$$Z = \sum_i e^{-\beta E_i}, \tag{1.2}$$

où $\beta = \frac{1}{k_b T}$, k_B est la constante de Boltzmann et T est la température. Dans la limite $T \to 0$, les seuls termes non nuls de la somme sont $e^{-\beta E_0}$, c'est-à-dire les termes

associés aux états fondamentaux du système. Par conséquent, pour un système dégénéré comportant N_0 états fondamentaux, la fonction de partition devient $Z = N_0$. Calculer la fonction de partition correspond ainsi à déterminer le nombre d'états fondamentaux.

Le développement du calcul quantique a aussi mené à de fructueux aboutissements. L'algorithme de comptage quantique, initialement introduit par Brassard, Høyer, Mosca et Tapp, tire avantage de l'algorithme de Grover et de l'algorithme d'estimation de phase quantique pour approximer le nombre de solutions en utilisant $(\sqrt{\frac{N}{M}})$ itérations, où M est le nombre de solutions, de taille exponentielle pour les problèmes SAT, et N est le nombre d'états possibles [42-44]. Le nombre d'itérations nécessaires profite ainsi d'un gain quadratique par rapport à la complexité optimale classique de $O(\frac{N}{M})$.

Notons que dans tous les cas, le comptage de solutions aux formules CNF, l'objet de ce travail, ne s'effectue jamais en temps polynomial. Même les approches approximatives nécessitent un nombre exponentiel d'opérations, indiquant davantage la difficulté de ce problème.

1.5 Transitions de phase

La complexité d'une instance aléatoire d'un problème NP n'est pas toujours équivalente. En réalité, certaines instances sont résolubles en temps polynomial, alors que d'autres sont intraitables. Les discussions autour des classes de complexité s'intéressent principalement à la complexité dans le pire des cas pour décrire la difficulté inhérente d'un problème. Cependant, certaines instances peuvent posséder une complexité inférieure, et donc être efficacement résolues par certains algorithmes.

Pour les problèmes SAT, la difficulté d'une formule CNF aléatoire est grandement dépendante au rapport $\alpha = \frac{m}{n}$, où m est le nombre de clauses et n est le nombre de variables. Intuitivement, cette difficulté provient de l'ajout de contraintes au problème sous la forme de clauses. Plus surprenamment, la complexité d'une telle instance suit une transition de phase nommée *transition de phase critique*. En

effet, il existe une valeur critique α_c où les instances du problème SAT passent de satisfaisables à insatisfaisables dans la limite asymptotique de n. La difficulté du problème SAT est maximale juste avant cette transition. Pourquoi est-ce le cas ?

Pour expliquer l'apparition de cette difficulté, il est nécessaire de donner un aperçu du paysage entourant cette transition. Plusieurs autres transitions de phase existent avant la transition critique, telle qu'illustré à la figure 1.2. Notons que la discussion suivante s'inspire de l'ouvrage de Moore et Mertens [45], une incroyable référence sur la théorie du calcul. Ces transitions, similairement aux transitions de phase du modèle d'Ising, correspondent à des changements dans l'organisation de l'ensemble des solutions. Pour décrire cette réorganisation, une mesure de similarité est nécessaire pour comparer les différentes entrées au problème SAT. La *distance de Hamming* entre deux chaînes de bits de même taille correspond au nombre de positions dans les chaînes où les bits correspondants diffèrent. L'espace de solutions se sépare typiquement en amas de solutions composé d'un nombre exponentiel de solutions, où chaque solution d'un amas est séparée par une large distance de Hamming avec les solutions des amas avoisinants. Pour le modèle d'Ising, une chaîne de Markov retournant les spins un à un nécessite un temps exponentiel pour passer d'un amas à un autre en raison de la présence d'états à haute énergie à la frontière de l'amas. La même situation apparait pour le problème SAT en remplaçant l'énergie par le nombre de contraintes non respectées.

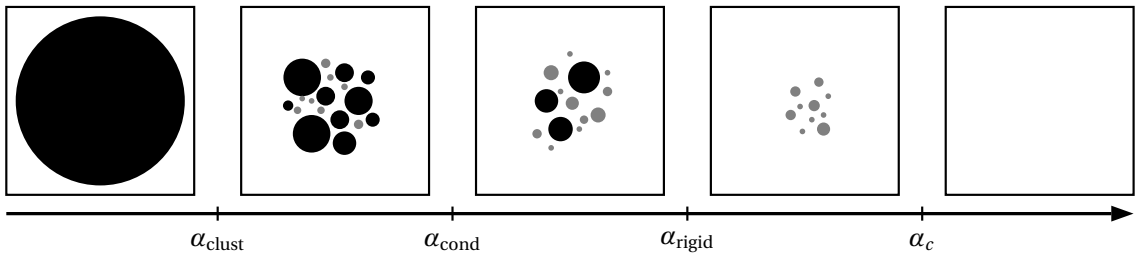

FIGURE 1.2 – Schéma des transitions de phase du problème SAT selon le rapport du nombre de clauses au nombre de variables α. Les transitions de phases illustrées sont la transition de regroupement α_{clust}, la transition de condensation α_{cond}, la transition de congélation α_{rigid} et la transition de satisfaisabilité α_c. Figure tirée de [45].

Avant la première transition de phase située à α_{clust}, la *transition de phase de regroupement*, toutes les solutions d'une instance du problème SAT sont situées au

sein du même amas. Après cette transition, un nombre exponentiel d'amas font leur apparition, séparés par une grande distance d'Hamming. Deux autres transitions apparaissent aussi en augmentant la densité des clauses sur les variables. Après la *transition de phase de condensation* à α_{cond}, un nombre sous-exponentiel d'amas domine l'ensemble de solutions. Finalement, les variables de chaque amas sont forcées de prendre des valeurs particulières après la *transition de phase de congélation* à α_{rigid}.

Les approches locales échouent à la transition de condensation en raison de la présence de corrélations à longues distances, comme mis en évidence par la méthode de propagation des convictions [46]. La propagation par sondage corrige la propagation des convictions en adressant les corrélations à longues distances et permet de localiser la transition de congélation suivant la transition de condensation [46]. Ces méthodes suggèrent que la difficulté du problème SAT provient de la transition de congélation.

La transition de phase critique pour NAE3SAT est située à $\alpha_c \approx 2.1$ [47] et à $\alpha_c \approx 2/3$ [48] pour 1-in-3SAT. Les problèmes #SAT semblent quant à eux être aussi maximalement complexes près de la phase de satisfaisabilité [49, 50]. Les instances du problème #1-in-3SAT près du seuil critique appartiennent à la catégorie des problèmes bloqués, où la distance de Hamming entre n'importe quelles solutions est de $O(n)$. Ceux-ci font partis des problèmes les plus difficiles de la classe #P [51].

L'approche utilisée dans le travail de ce mémoire se base sur l'optimisation des paramètres de circuits quantiques. Cette approche n'est pas locale comme avec les algorithmes classiques typiquement utilisés, ce qui laisse ainsi espérer un possible avantage. Par exemple, le problème #XORSAT est résoluble avec une méthode globale, l'élimination gaussienne, en temps polynomial [45].

Chapitre 2

Échantillonnage quasi uniforme et comptage approximatif randomisé

Les problèmes de décision, ou d'existence, et de comptage tels qu'introduits dans le chapitre précédent ne forment qu'une infime partie des problèmes étudiés dans la théorie de la complexité. Ces problèmes se cadrent notamment parmi les problèmes suivants :

(1) **Existence** : Existe-t-il une solution au problème ?
(2) **Construction** : Construire une solution au problème.
(3) **Génération uniforme** : Générer uniformément et aléatoirement une solution au problème.
(4) **Comptage** Combien de solutions satisfont le problème ?

Il n'est pas évident de prime abord de comparer la complexité de ces problèmes. Toutefois, dans une publication marquante pour le domaine de la complexité du comptage, Jerrum, Valiant et Vazirani offrent une unique perspective sur l'écart de complexité entre ces différents problèmes [3]. Les auteurs suggèrent d'abord que la génération uniforme est strictement plus difficile que la construction, et donc que l'existence comme la construction est évidemment au moins aussi difficile que l'existence. De plus, ils avancent que la génération uniforme est plus simple que le comptage, tout en offrant une réduction de la génération uniforme au comptage approximatif. En relaxant le comptage approximatif au comptage approximatif

randomisé, ils établissent une correspondance entre ce dernier et la génération quasi uniforme.

Ce travail éclaircit la compréhension de la théorie de la complexité, particulièrement pour la génération uniforme et du comptage. La dernière correspondance entre le comptage approximatif randomisé et la génération quasi uniforme s'avère particulièrement intéressante et forme la base du travail de ce mémoire. Cette interréduction mène en effet à un algorithme de comptage approximatif résolvant approximativement certains problèmes de comptage, possédant la propriété d'auto-réductibilité, à l'aide d'un générateur de solutions quasi uniforme. Par souci de commodité, cet algorithme est appelé l'*algorithme de JVV* selon le nom de ses auteurs. Parmi ses principaux attraits est sa capacité à obtenir un compte approximatif avec un nombre polynomial d'appels au générateur de solutions, sous la condition que ce dernier soit génère une distribution de solutions suffisamment uniforme. Cette propriété s'avère particulièrement utile pour les problèmes possédant un nombre exponentiel de solutions, comme le problème SAT.

Les trois concepts nécessaires à l'algorithme de JVV sont formalisés dans cette section : l'auto-réductibilité à la section 2.1, l'échantillonnage quasi uniforme à la section 2.2 et le comptage approximatif randomisé à la section 2.3. Pour conclure, la section 2.4 présente une description en profondeur de l'algorithme de JVV.

2.1 Auto-réductibilité

Avant d'introduire le concept d'auto-réductibilité, il est utile de définir une notion supplémentaire. Les problèmes abordés dans l'introduction de ce chapitre concernent généralement des objets mathématiques discrets, tels que les formules propositionnelles, les graphes, les permutations, et bien plus. Ces objets se regroupent sous une même ombrelle, nommée *structure combinatoire*, définie comme un système discret fini comportant des éléments et des relations bien définies. Par exemple, une formule propositionnelle, composée de variables booléennes et de contraintes sous forme d'opérateurs booléens, est associée à un ensemble discret de solutions. Conséquemment, le domaine de l'*énumération combinatoire* cherche à déterminer le nombre d'éléments satisfaisant la relation. Pour le problème SAT, cela

consiste à trouver le nombre d'assignations satisfaisant la formule propositionnelle. De manière similaire, l'*optimisation combinatoire* tente de trouver la meilleure solution possible au problème.

L'*auto-réductibilité* (« self-reducibility ») est un concept complexe, essentiel à la compréhension du calcul et de la complexité, découlant de l'introduction de la *réduction automatique* (« autoreducibility ») [52, 53]. Ces concepts sont particulièrement importants dans le contexte de génération aléatoire et du comptage approximatif, mais aussi pour la réduction entre le problème de décision et les problèmes de recherche.

Définition 2.1 – Réduction automatique

Un problème algorithmique est dit *automatiquement réductible* s'il peut être résolu par un algorithme résolvant d'autres instances du même problème, sans que l'algorithme puisse interroger l'instance particulière qu'il cherche à résoudre.

Les problèmes automatiquement réductibles contiennent de l'information d'appartenance redondante, c'est-à-dire qu'il existe une structure dans l'ensemble de problèmes pouvant être exploitée pour simplifier le calcul d'une instance donnée. Ainsi, un algorithme peut résoudre une instance en utilisant l'information redondante présente dans d'autres instances, évitant ainsi les requêtes directes à l'instance en question. Connaître la solution à un autre problème peut alors simplifier la résolution du problème initial.

Pour discuter de la génération aléatoire et le comptage approximatif, l'*auto-réductibilité descendante*, une forme limitée de la réduction automatique, est une définition plus adéquate. En effet, cette condition est nécessaire à l'application de l'algorithme JVV.

Définition 2.2 – Auto-réductibilité descendante

Un problème algorithmique est dit *auto-réductible descendant* s'il peut être résolu grâce à un algorithme résolvant des instances de taille strictement inférieure.

Cette propriété s'éclaircit en prenant le problème SAT comme exemple. L'auto-réductibilité appliquée au problème de satisfaisabilité s'exprime facilement avec la

relation suivante [54] :

Relation 2.1 – Auto-réductibilité pour les problèmes SAT

Soit une constante $n \geq 1$ et un problème SAT décrit par la formule propositionnelle $\varphi(x_1, x_2, \ldots, x_n)$ où $x_i \in \{0, 1\}$. Alors,

$$\varphi(x_1, x_2, \ldots, x_n) = 1 \iff \varphi(x_1 = 0, x_2, \ldots, x_n) = 1 \vee \varphi(x_1 = 1, x_2, \ldots, x_n) = 1$$

Cette relation implique que l'ensemble de solutions d'une instance donnée peut être exprimé comme l'ensemble de solutions de deux instances plus petites du problème. Supposons que l'on souhaite résoudre une instance du problème SAT décrit par la formule CNF $\varphi(x_1, x_2, \ldots, x_n)$. Soit $\varphi_0 = \varphi(x_1 = 0, \ldots, x_n)$ et $\varphi_1 = \varphi(x_1 = 1, \ldots, x_n)$ deux sous-instances de l'instance φ, où la variable x_1 est remplacée par 0 et 1 respectivement. Ce faisant, la formule φ est alors raccourcie, comme certaines clauses sont satisfaites ou du moins réduites. Pour que l'instance φ soit satisfaisable, il est nécessaire qu'au moins une des sous-instances φ_0 et φ_1 soit satisfaisable. Dans le cas contraire, l'instance originale φ ne peut être satisfaisable, car il n'existe pas de solutions peu importe la valeur de x_1. Ainsi, il suffit de considérer la disjonction des sous-problèmes possibles du problème original pour résoudre ce dernier. Les sous-problèmes obtenus étant aussi des formules propositionnelles, ceux-ci peuvent aussi être décomposés en problème de taille inférieure récursivement.

Notons qu'il n'est pas nécessaire de construire la relation 2.1 avec la variable x_1, n'importe quelle variable x_i peut aussi être utilisée. Notons aussi que le problème SAT est décrit comme une auto-réductibilité à longueur décroissante 2-disjonctive. Ici, 2-disjonctive fait référence à une formule propositionnelle composée d'une disjonction de conjonctions de deux variables au plus. Une longueur décroissante ou descendante signifie que l'algorithme résout des instances de taille strictement inférieure.

La relation 2.1 exemplifie une deuxième façon de voir l'auto-réductibilité : la résolution partielle d'une instance d'un problème laisse une plus petite instance du même type de problème. Une interprétation alternative est qu'une instance peut être résolue en résolvant des plus petites instances et en assemblant les sous-instances

ensemble, tel l'assemblage d'un casse-tête.

Afin d'obtenir une meilleure perspective sur la notion d'auto-réductibilité, il est pertinent d'exprimer la structure d'une relation auto-réductible φ sous la forme d'un arbre orienté, nommé *arbre d'auto-réductibilité*, illustré à la figure 2.1. Dans cet arbre, les sommets représentent à la fois une chaîne de bits w de taille m et une instance de problème φ_w, où φ_w représente la sous-instance du problème φ dont les premières variables sont remplacées par la chaîne de bits w. Le sous-problème est alors dit *fixé* par le préfixe w. Les arêtes du graphe représentent l'assignation d'une variable. La racine de l'arbre correspond à l'instance du problème initial φ ainsi qu'à une solution partielle nulle. Les enfants de cette racine sont par la suite donnés par les sous-instances φ_w pour tous les préfixes w de taille 1 possibles. Le reste de l'arbre est défini récursivement de la même manière en augmentant la taille m de la chaîne de bits w à chaque niveau jusqu'aux feuilles de l'arbre. Ces feuilles représentent finalement une entrée au problème φ, étant soit une solution ou une non-solution au problème.

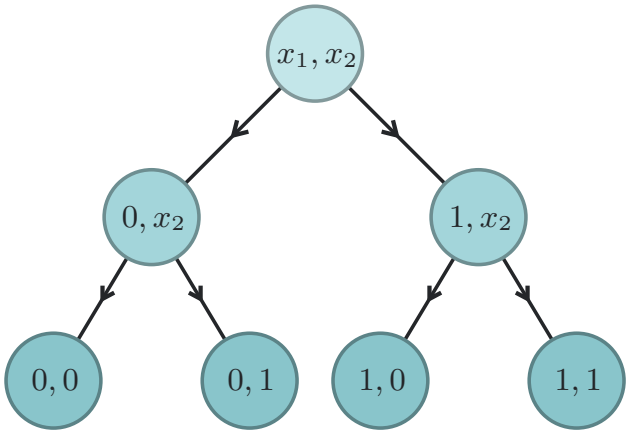

FIGURE 2.1 – Un exemple de l'arbre d'auto-réductibilité pour une formule CNF $\varphi(x_1, x_2)$. Les sommets représentent des attributions de littéraux dans φ alors que les arêtes orientées indiquent l'affectation d'un littéral.

L'auto-réductibilité est simplement généralisable aux problèmes dont les variables peuvent prendre d valeurs, dit auto-réductibilité d-disjonctive, en prenant la disjonction sur les d valeurs possibles. Le degré de l'arbre est en conséquence donné par d. Notons aussi que l'arbre d'auto-réductibilité est parfois défini tel que

les feuilles correspondent uniquement à des solutions au problème auto-réductible. Dans ce cas, les non-solutions ne sont pas incluses dans les ramifications de l'arbre.

Par complétude, une définition rigoureuse de l'auto-réductibilité dans le sens de Schnorr [55] est donnée en suivant les explications de Jerrum, Valiant et Vazirani [3]. Celle-ci permet l'introduction d'un langage pertinent pour le reste de ce chapitre.

Définition 2.3 – Auto-réductibilité descendante

Soit Σ^* un ensemble fixe et fini encodant les instances d'un problème ainsi que leurs solutions. Soit $R \subseteq \Sigma^* \times \Sigma^*$ une relation binaire assignant à chaque instance de problème $x \in \Sigma^*$ un ensemble de solutions $\{\, y \in \Sigma^* : xRy \,\}$. Une relation $R \subseteq \Sigma^* \times \Sigma^*$ est auto-réductible si et seulement si

(1) il existe une fonction calculable en temps polynomial $g \in \Sigma^* \to \mathbb{N}$ tel que $xRy \implies |y| = g(x)$;

(2) il existe une fonction calculable en temps polynomial $\psi \in \Sigma^* \times \Sigma^* \to \Sigma^*$ et $\sigma \in \Sigma^* \to \mathbb{N}$ satisfaisant

$$\begin{aligned} &\sigma(x) = O(\log |x|)\,, \\ &g(x) > 0 \Rightarrow \sigma(x) > 0 \quad \forall x \in \Sigma^\star\,, \\ &|\psi(x,w)| \leqslant |x| \quad \forall x, w \in \Sigma^\star\,, \end{aligned}$$

et tel que, pour tout $x \in \Sigma^*$ et $y = y_1 \dots y_n \in \Sigma^*$,

$$(x, y_1 \dots y_n) \in R \Leftrightarrow \left(\psi\left(x, y_1 \dots, y_{\sigma(x)}\right), y_{\sigma(x)+1} \dots y_n\right) \in R\,.$$

Démystifions cette définition. L'ensemble Σ est un alphabet fini encodant à la fois les instances du problème et ses solutions. Le problème SAT est un exemple de relation binaire R, où x encode une formule booléenne et y encode une assignation satisfaisable parmi l'ensemble des entrées possibles Σ^* tel que

$$\begin{aligned} R = \{(x,y) : x \in \Sigma^* \text{ encode une formule booléenne } B, \\ y \in \Sigma^* \text{ est une assignation satifaisable de } B\}\,. \end{aligned} \tag{2.1}$$

La notation xRy signifie que y est une solution valide à l'instance x. La première condition implique que la taille des solutions, donnée par la fonction g, est bornée par

une fonction calculable en temps polynomial. La deuxième condition implique que, pour une instance x et un préfixe w de taille $\sigma(x)$ de n'importe quelle solution y au problème x, la fonction ψ donne une instance x' dont les solutions sont exactement celles qui, lorsque concaténées avec w, forment les solutions de x. La fonction σ donne alors la granularité des solutions, c'est-à-dire le nombre de caractères de y utilisés pour réduire l'instance x. Plus précisément, $\psi(x, w) \leq |x|$ assure que la taille du problème diminue à mesure que la réduction se poursuit. L'implication $g(x) > 0 \implies \sigma(x) > 0$ signifie que la réduction continue tant que la taille de l'instance x est non nulle. La dernière relation énonce que la résolution du problème pour $\langle x, y \rangle$ peut être réduite à la résolution d'instances de taille inférieure, c'est-à-dire que $(x, y_1 \dots y_n)$ est reliée à $(\psi(x, y_1 \dots y_{\sigma(x)}), y_{\sigma(x)+1} \dots y_n)$.

Tous les problèmes NP-complet sont auto-réductibles [56], mais ce n'est pas le cas pour tous les problèmes de la classe NP [57]. De nombreux problèmes NP sont aussi auto-réductible descendant. Ces observations impliquent que l'algorithme de JVV s'applique en théorie à de nombreux problèmes intéressants. Toutefois, trouver une relation auto-réductible en pratique n'est pas toujours évident.

La définition de l'auto-réductibilité utilisée dans cette section s'applique principalement à l'équivalence entre l'échantillonnage quasi uniforme et le comptage approximatif randomisé. Cependant, une définition alternative [56] est souvent introduite pour l'équivalence entre les problèmes de décision et les problèmes de recherche, où un problème de recherche ne demande pas seulement de montrer l'existence d'une solution, mais de construire une telle solution. Si un problème est auto-réductible à ce sens, alors le problème de recherche est de la même complexité que le problème de décision.

2.2 Échantillonnage quasi uniforme

L'*échantillonnage uniforme* de structures combinatoires consiste à générer aléatoirement des solutions de manière uniforme, c'est-à-dire avec une probabilité suivant la distribution de probabilité uniforme des solutions, à un problème donné. Le problème de génération uniforme est assez méconnu, mais possède plusieurs

applications, dont la construction d'éléments représentatifs d'un ensemble, la formulation de conjectures pour un ensemble ou la vérification fonctionnelle du matériel informatique [58, 59].

La génération uniforme étant une demande plutôt rigide, il est parfois nécessaire de relaxer la condition d'uniformité pour une quasi-uniformité. Une distribution quasi uniforme est en pratique indiscernable d'une distribution uniforme, mais ce relâchement facilite certaines preuves. La *distance en variation totale* est une mesure de distance statistique définie entre deux distributions de probabilité, déterminant la similitude entre ces distributions. Cette mesure permet, entre autres, de déterminer dans quelle mesure une distribution quasi uniforme se rapproche de la distribution uniforme.

Définition 2.4 – Distance en variation totale

Soit deux distributions de probabilité P et Q définies sur un ensemble dénombrable χ. La distance en variation totale est

$$\|P - Q\|_{TV} \equiv \frac{1}{2} \sum_{x \in \chi} |P(x) - Q(x)| .$$

Une distribution est alors dite quasi uniforme si sa distance en variation totale avec la distribution uniforme est suffisamment petite. Une définition commune demande que cette distance diminue proportionnellement avec l'inverse de la taille du problème. Un *échantillonneur quasi uniforme de tolérance* ξ génère alors des solutions selon une distribution de probabilités où la distance en variation totale entre celle-ci et la distribution uniforme est plus petite que ξ [60].

Définition 2.5 – Échantillonneur quasi uniforme

Un échantillonneur quasi uniforme pour une relation binaire $R \subseteq \Sigma^* \times \Sigma^*$ assignant à chaque instance d'un problème $x \in \Sigma^*$ un ensemble de solutions $S(x) = \{ y \in \Sigma^* : xRy \}$, est un algorithme aléatoire prenant en entrée une instance x et une tolérance d'échantillonnage $\xi > 0$ et générant une solution $y \in S(x)$ tel que

$$\|Y - U\|_{TV} \leq \xi \,,$$

où Y est la distribution de probabilité de y et U est la distribution de probabilité uniforme sur $S(x)$. Si l'algorithme s'exécute en temps borné par une fonction polynomiale en $|x|$ et en $\ln(\xi^{-1})$, on parle d'échantillonneur quasi uniforme pleinement polynomial (« Fully Polynomial Almost Uniform Sampler») (FPAUS).

Une notion supplémentaire est introduite dans ce mémoire afin de simplifier la notation. La *non-uniformité*, de manière similaire à la distance en variation totale, décrit la distance entre une distribution de probabilité P et la distribution de probabilité uniforme U.

Définition 2.6 – Non-uniformité

Soit la distribution de probabilité P et la distribution de probabilité uniforme U telles que $U(x) = 1/|x|$ définies sur un ensemble dénombrable χ. La non-uniformité est

$$\eta \equiv \frac{1}{2} \sum_{x \in \chi} |P(x) - U(x)| \,.$$

Un échantillonneur quasi uniforme de tolérance ξ possède en conséquence une non-uniformité plus petite que ξ.

2.3 Comptage approximatif randomisé

Comme mentionné au chapitre 1, les problèmes de comptage sont d'une grande complexité et nécessitent en pratique des méthodes approximatives pour la résolution de problèmes de grandes tailles. Le *comptage approximatif randomisé* simplifie davantage les attentes en ne demandant une solution approximative valide qu'avec une certaine probabilité. Un *schéma d'approximation randomisé de tolérance* ε se définit de manière similaire à l'algorithme d'approximation pour l'optimisation, décrit à la section 1.3.

Définition 2.7 – Schéma d'approximation randomisé

Un schéma d'approximation randomisé pour un problème de comptage $f : \Sigma^* \to \mathbb{Q}$ est un algorithme randomisé prenant en entrée une instance d'un problème $x \in \Sigma^*$ et une tolérance d'erreur $\varepsilon > 0$ et qui génère un nombre rationnel $N \in \mathbb{Q}$ tel que, pour toute instance x,

$$\Pr\left[(1+\varepsilon)^{-1} f(x) \leq N \leq (1+\varepsilon) f(x)\right] \geq \frac{3}{4}.$$

Si l'algorithme s'exécute en temps borné par une fonction polynomiale en $|x|$ et ε^{-1}, alors on parle de schéma d'approximation randomisé pleinement polynomial (« Fully Polynomial Randomized Approximation Scheme ») (FPRAS).

La valeur $3/4$ est un nombre arbitraire pouvant être remplacée par n'importe quel nombre dans l'intervalle ouvert $(\frac{1}{2}, 1)$. Ce nombre, représentant la confiance de succès du schéma, peut être augmenté à n'importe quelle valeur $1 - \delta$ pour $\delta > 0$ au coût d'un facteur multiplicatif $O(\log(\delta^{-1}))$. Pour ce faire, il suffit de répéter le schéma original $O(\log(\delta^{-1}))$ fois et de prendre la médiane des résultats obtenus.

2.4 Algorithme de Jerrum-Valiant-Vazirani

Le travail de Jerrum, Valiant et Vazirani [3] (voir aussi [61, 62]) établit une correspondance entre l'échantillonnage quasi uniforme et le comptage approximatif randomisé. Pour montrer celle-ci, les auteurs présentent deux algorithmes permettant le passage d'un problème à l'autre. Bien que la réduction du comptage à l'échantillonnage est intéressante, elle ne sera discutée que brièvement à la fin de la section comme la réduction d'importance est ici son inverse. L'algorithme de comptage approximatif randomisé, surnommé *algorithme de JVV*, permet de trouver le nombre de solutions d'un problème auto-réductible de la classe #P à une erreur multiplicative près avec un nombre polynomial d'appels à un générateur de solution quasi uniforme. Le comptage approximatif randomisé est utilisé plutôt que le comptage approximatif en raison de l'impossibilité d'obtenir un résultat déterministe à partir d'une procédure aléatoire.

Théorème 2.1 – Algorithme de JVV

Si un problème auto-réductible dans #P de taille n admet un générateur de solutions avec une non-uniformité $\eta = O(1/n)$, alors le nombre de solutions peut être approximé à une erreur multiplicative $O(\eta n)$ avec une haute probabilité en utilisant un nombre polynomial d'appels au générateur.

En exigeant un échantillonneur quasi uniforme pleinement polynomial (FPAUS), l'algorithme de JVV fournit un schéma d'approximation randomisé pleinement polynomial (FPRAS) pour le comptage approximatif de relations auto-réductibles. Plus précisément, pour un problème de n variables, $O(\frac{n^2 \log \delta^{-1}}{\varepsilon^2})$ échantillons sont nécessaires pour obtenir un compte approximatif de tolérance ε et de confiance δ.

Comment est-ce que l'algorithme de JVV opère ? Commençons par introduire l'idée principale derrière l'algorithme pour ensuite donner un exemple simple et compléter avec une description plus formelle.

Pour mieux comprendre son exécution, l'arbre d'auto-réductibilité, présenté à la section 2.1, est employé. L'algorithme de JVV implique de se déplacer vers le haut ou vers le bas de cet arbre, en s'intéressant à toutes les sous-instances en chemin. Supposons qu'il soit possible d'échantillonner uniformément les solutions de

n'importe quelle instance de taille n du problème SAT (la généralisation à un échantillonneur quasi uniforme est effectuée plus bas). Commençons par échantillonner des solutions au problème initial φ représenté par la racine de l'arbre. Soit P_0 et P_1 les probabilités que la première variable x_1 soit respectivement de 0 et 1, calculées à partir des solutions échantillonnées. Ces probabilités reflètent la proportion de solutions dans chaque sous-arbre. Notons la valeur la plus probable par w_1. Descendons l'arbre vers l'enfant le plus probable φ_{w_1}, représentant la formule φ réduite par le préfixe w_1, en gardant en mémoire la probabilité P_{w_1}. Comme l'échantillonneur peut aussi résoudre le problème subséquent, nous pouvons encore déterminer la valeur la plus probable w_2 de la variable suivante x_2 à partir de la probabilité $P_{w_1 w_2}$. Le processus est répété jusqu'à atteindre les feuilles de l'arbre, en sauvegardant les probabilités $P_{w_1 w_2 \dots w_i}$ le long du chemin $w_1 w_2 \dots w_i$ choisi. Au dernier noeud avant les feuilles, nous obtenons une solution donnée par $w_1 w_2 \dots w_n$ avec une probabilité $P_{w_1 w_2 \dots w_n}$. Comme nous sommes à la base de l'arbre, cette probabilité indique le nombre de solutions que possède la dernière sous-instance. En effet, ce sous-arbre contient nécessairement $\frac{1}{P_{w_1 w_2 \dots w_n}}$ solutions, car il y a nécessairement une ou deux solutions à la dernière sous-instance. En remontant l'arbre, on remarque que la sous-instance précédente contient $\frac{1}{P_{w_1 w_2 \dots w_{n-1}}}$ fois plus de solutions. Par conséquent, le nombre de solutions dans cette partie de l'arbre est de $\frac{1}{P_{w_1 w_2 \dots w_n}} \cdot \frac{1}{P_{w_1 w_2 \dots w_{n-1}}}$. Continuant cette procédure jusqu'à la racine, nous trouvons que le nombre de solutions au problème φ est donné par

$$N = \frac{1}{P_{w_1}} \cdot \frac{1}{P_{w_1 w_2}} \cdots \frac{1}{P_{w_1 w_2 \dots w_n}} = \frac{N}{N_{w_1}} \cdot \frac{N_{w_1}}{N_{w_1 w_2}} \cdots \frac{N_{w_1 w_2 \dots w_{n-1}}}{N_{w_1 w_2 \dots w_n}}, \tag{2.2}$$

où $N_{w_1 \dots w_i}$ est le nombre de solutions au problème $\varphi_{w_1,\dots,w_i}$. En exprimant les probabilités sous la forme de rapport entre le nombre de solutions d'un plus petit sous-arbre au sein d'un sous-arbre, comme montré à l'équation 2.2, nous remarquons que tous les termes s'annulent à l'exception de N et de $N_{w_1 w_2 \dots w_n}$. Ce dernier étant nécessairement 1, la multiplication de l'inverse des probabilités donne bien le nombre de solutions au problème.

Éclaircissons maintenant cette idée à l'aide d'un exemple simple illustré à la figure 2.2. Soit une instance du problème SAT décrite par la formule CNF $\varphi(x_1, x_2) = \neg x_1 \wedge x_2$, dont les solutions sont les couples $(0,0)$, $(0,1)$ et $(1,1)$. Un arbre est

construit de manière à représenter toutes les sous-instances de l'instance originale. La racine de l'arbre représente le problème initial φ et ses deux variables x_1 et x_2. La première couche représente les sous-problèmes φ_0 et φ_1 où la première variable du problème φ est remplacée par 0 et 1 respectivement. Les feuilles représentent les assignations possibles au problème φ, c'est-à-dire 00, 01, 10 et 11, comme toutes les variables sont fixées à une certaine valeur. Les arêtes de l'arbre décrivent la probabilité de l'assignation des variables. Par exemple, la probabilité $P_{11} = 1$ indique que les échantillons générés à partir du sous-problème φ_1 commencent par le préfixe 11 avec une probabilité 1. Maintenant que l'arbre d'auto-réductibilité est construit, comptons le nombre de solutions à la formule $\varphi(x_1, x_2)$. En échantillonnant l'instance originale, nous trouvons que les échantillons possèdent une probabilité $P_0 = 2/3$ de commencer avec la variable $x_1 = 0$. La sous-instance $\varphi_0 = \varphi(0, x_1)$ est donc choisie, comme il s'agit de l'enfant le plus probable, et de nouvelles solutions sont générées. Nous remarquons que celles-ci possèdent la même probabilité $P_{00} = P_{01} = \frac{1}{2}$ de commencer par le préfixe 00 que par le préfixe 01. Après avoir choisi le préfixe 01, le compte est obtenu avec $N = (P_0 \cdot P_{01})^{-1} = (\frac{2}{3} \cdot \frac{1}{2})^{-1} = 3$.

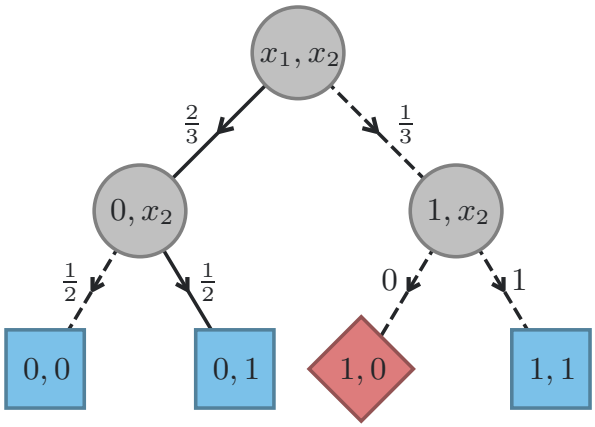

FIGURE 2.2 – Un exemple de l'arbre d'auto-réductibilité dans l'algorithme JVV pour une formule CNF $\varphi(x_1, x_2)$. Les sommets représentent des attributions de littéraux dans φ, avec les solutions représentées par des carrés bleus et les non-solutions par des losanges rouges. Les arêtes orientées indiquent l'affectation d'un littéral ainsi que sa probabilité associée. En suivant le chemin des probabilités maximales, illustré par une ligne pleine, le compte est obtenu avec $N = (P_0 \cdot P_{01})^{-1} = 3$.

Quelques notes sont ici nécessaires. Le chemin des probabilités maximales est

choisi, car une meilleure précision est atteinte avec un plus grand nombre de différentes solutions échantillonnées. De plus, ce choix évite d'obtenir une probabilité nulle, ce qui arrive si le sous-arbre ne possède aucune solution. Jusqu'à maintenant, le générateur a été considéré comme uniforme. Il est cependant possible d'utiliser une distribution quasi uniforme, qui introduit une petite erreur additionnelle pouvant être absorbée dans la tolérance ε.

Maintenant que l'intuition derrière l'algorithme de JVV est éclaircie, décrivons plus formellement celui-ci. Supposons que le nombre exact de solutions pour une instance de taille n du problème SAT est de N. Notons $x_{:m}$ le préfixe de taille m de la chaîne de bits x et $N_{x_{:m}}$ le nombre de solutions commençant avec le préfixe $x_{:m}$. Par la suite, supposons que nous avons accès à un générateur qui échantillonne l'ensemble de solutions uniformément et aléatoirement, tel que chaque solution est échantillonnée avec une probabilité $p^\star = \frac{1}{N}$. Pour n'importe quelle solution z, on peut écrire

$$\begin{aligned} p^\star = \frac{1}{N} &= \frac{N_{z_{:1}}}{N} \cdot \frac{N_{z_{:2}}}{N_{z_{:1}}} \cdots \frac{N_{z_{:n}}}{N_{z_{:n-1}}} \\ &= p(z_{:1}) \cdot p(z_{:2}|z_{:1}) \cdots p(z_{:n}|z_{:n-1}) \\ &= p(z_{:1}) \prod_{i=1}^{n-1} p(z_{:i+1}|z_{:i}) \,, \end{aligned} \tag{2.3}$$

où $N_{z_{:n}} \equiv 1$. La probabilité conditionnelle $p(z_{:i+1}|z_{:i}) \equiv N_{z_{:i+1}} / N_{z_{:i}}$ est la probabilité qu'une solution échantillonnée z' soit $z'_{i+1} = z_{i+1}$, sachant que $z'_{:i} = z_{:i}$. L'algorithme de JVV pour le comptage approximatif retourne une approximation à un facteur multiplicatif près de $p^\star$, et donc de N, en approximant les probabilités conditionnelles $p(z_{:i+1}|z_{:i})$. Cela est fait en générant un ensemble d'échantillons S et en approximant les probabilités conditionnelles avec

$$\tilde{p}(z_{:i+1}|z_{:i}) = \frac{\tilde{N}_{z_{:i+1}}}{\tilde{N}_{z_{:i}}} \,, \tag{2.4}$$

où $\tilde{N}_{z_{:i}}$ est le nombre de solutions dans l'ensemble d'échantillons commençant par $z_{:i}$. Donc,

$$\tilde{p}^\star = \tilde{p}(z_{:1}) \prod_{i=1}^{n-1} \tilde{p}(z_{:i+1}|z_{:i}) \approx \frac{1}{N} \,. \tag{2.5}$$

Le nombre approximatif de solutions est ainsi obtenu. La preuve donnée dans le

travail de Jerrum, Valiant et Vazirani montre qu'un ensemble d'échantillons de taille polynomiale est suffisant pour obtenir une approximation multiplicative au nombre de solutions exact [3]. L'algorithme se résume plus succinctement par l'algorithme 2.4.

Algorithme 1 : Algorithme de JVV

Entrée: Formule CNF : $f(n, m)$, Nombre d'échantillons : n_s

$w \leftarrow$ " "
$\tilde{N} \leftarrow 1$
pour $i \in \{1, \ldots, n\}$ **faire**
  $S \leftarrow \{\ \}$
  tant que $|S| < n_s$ **faire**
    $m \leftarrow$ Échantillonnage$(f(n, m))$
    $S \leftarrow S \cup \{\, m \,\}$
  fin tant que
  $w, \tilde{p} \leftarrow \arg\max_{w' \in \{\, w+0, w+1 \,\}} |\{\, s \in S : s_{:|w'|} = w' \,\}|/|S|$
  $\tilde{N} \leftarrow \tilde{N}/\tilde{p}$
fin pour
retourne $\tilde{N}$

Finalement, un générateur de solutions aléatoire uniforme peut aussi être construit à partir d'un compteur approximatif en inversant la procédure présentée ci-haut. Pour ce faire, descendons l'arbre d'auto-réductibilité d'un problème φ avec N solutions à nouveau. À chaque noeud de l'arbre $\varphi_{w_1 w_2, \ldots w_i}$, comptons approximativement le nombre de solutions à ses deux enfants $\varphi_{w_1 w_2 \ldots w_i 0}$ et $\varphi_{w_1 w_2 \ldots w_i 1}$ et empruntons une des deux branches avec une probabilité proportionnelle au nombre de solutions dans les sous-arbres, c'est-à-dire $\frac{N_{w_1 w_2 \ldots w_{i+1}}}{N_{w_1 w_2 \ldots w_i}}$. En suivant cette procédure jusqu'aux racines, une solution est obtenue avec une probabilité $\frac{1}{N}$.

Chapitre 3

Algorithmes variationnels quantiques

En général, le comptage est un problème ardu. En acceptant une solution approximative, la complexité de ce problème peut être déplacée à l'échantillonnage quasi uniforme de solutions grâce à l'algorithme de JVV. Toutefois, la construction d'un tel générateur n'a pas encore été évoquée. En fait, la majorité des solveurs de problèmes #P ne se basent pas sur l'échantillonnage en raison de la difficulté d'obtenir une distribution uniforme composée de solutions. Est-ce que le calcul quantique peut offrir une méthode efficace pour la génération uniforme de solutions ?

Les *algorithmes variationnels quantiques* (« Variational Quantum Algorithms ») (VQA) sont des algorithmes hybrides, c'est-à-dire composés d'une partie quantique et d'une partie classique, conçus pour exploiter les avantages du calcul quantique tout en profitant de la puissance des algorithmes classiques [2]. Ces algorithmes ont émergé comme la stratégie dominante pour atteindre l'avantage quantique avec le matériel informatique quantique actuel, connu sous le nom des ordinateurs quantiques bruités de taille intermédiaire (« Noisy Intermediate-Scale Quantum ») (NISQ). En effet, les algorithmes quantiques possédant un avantage par rapport aux algorithmes classiques sont actuellement hors d'atteinte pour les ordinateurs quantiques du moment en raison de la taille des systèmes nécessaires et des erreurs causées par le bruit. Les VQA s'inspirent des méthodes d'apprentissage automatique pour résoudre des problèmes d'optimisation combinatoire avec des circuits de faible profondeur sans se soucier de la correction des erreurs des qubits. Un état quantique initial facile à préparer est évolué unitairement avec un circuit quantique

paramétré et la valeur moyenne d'une fonction de coût est estimée par de multiples mesures du circuit dans une base appropriée. Les paramètres du circuit sont alors ajustés itérativement par un optimiseur classique afin de minimiser la fonction de coût et ainsi préparer un état près d'une superposition des solutions du problème. Cette approche limite les inconvénients causés par le bruit en raison de l'utilisation de circuits paramétrés limitant la taille des circuits utilisés. La superposition des solutions préparée par les VQA peut être utilisée pour la composante d'échantillonnage de l'algorithme de JVV, améliorant potentiellement les méthodes de comptage basées sur l'échantillonnage.

Parmi les algorithmes englobés par les VQA se place le célèbre *algorithme quantique d'optimisation approximative* (« Quantum Approximate Optimization Algorithm ») (QAOA) [5]. Étant l'un des premiers VQA appliqués aux problèmes d'optimisation combinatoire, cet algorithme se prête particulièrement bien à nos demandes. De plus, un grand nombre de travaux ont caractérisé celui-ci en profondeur, facilitant ainsi son application à la question directrice du travail de ce mémoire. Cet algorithme est conséquemment l'objet principal de ce chapitre.

L'algorithme adiabatique quantique, le fondement théorique de QAOA, est d'abord introduit à la section 3.1. Après avoir décrit en détail QAOA à la section 3.2, différentes variantes de QAOA sont explorées, telles sa généralisation à l'ansatz quantique à opérateurs alternants et une variante de ce dernier employant le forçage de Grover à la section 3.3. Finalement, deux propriétés de QAOA sont explorées, c'est-à-dire l'initialisation et l'optimisation des paramètres du circuit quantique à la section 3.4 et le biais d'échantillonnage à la section 3.5.

3.1 Algorithme adiabatique quantique

Le *théorème adiabatique*, introduit par Born et Fock [63], peut être énoncé simplement comme suit :

Théorème 3.1 – Théorème adiabatique

Un système physique demeure dans son état propre instantané si une perturbation donnée agit sur lui suffisamment lentement et s'il y a un intervalle significatif entre la valeur propre et le reste du spectre de l'hamiltonien.

Bien que différentes versions de ce théorème furent rigoureusement formulées [64], une version approximative de celui-ci, proposée par Messiah [65] et rectifiée par Amin [66], est présentée ici dans l'objectif d'élucider les mécanismes du théorème. Un système quantique, décrit par un hamiltonien dépendant du temps $H(t)$, évolue selon l'équation de Schrödinger

$$i\hbar \frac{\partial \left|\psi(t)\right\rangle}{\partial t} = H(t) \left|\psi\right\rangle \ . \tag{3.1}$$

Considérons ici que l'hamiltonien $H(t)$ peut s'écrire sous la forme $H(t) = \tilde{H}(s)$, où $s = t/T \in [0,1]$ est le temps adimensionnel, de manière que T contrôle le taux de variation dans le temps de $H(t)$. Soit $|\varepsilon_j(s)\rangle$ les états propres instantanés (potentiellement dégénérés) de $\tilde{H}(s)$ avec énergie ε_j tels que

$$\tilde{H}(s) \left|\varepsilon_j(s)\right\rangle = \varepsilon_j(s) \left|\varepsilon_j(s)\right\rangle \ , \tag{3.2}$$

où $\varepsilon_j(s) < \varepsilon_{j+1}(s)\ \forall j, s$ et $j \in \{0, 1, 2, \dots\}$. L'approximation adiabatique indique qu'un état initial préparé dans un des états propres instantanés $|\varepsilon_j(0)\rangle$ demeure dans le même état propre instantané $|\varepsilon_j(t)\rangle$ à une phase globale près, à condition que $\varepsilon_i(s) - \varepsilon_j(s) \neq 0$ et

$$T \gg \max_{s\in[0,1]} \frac{|\langle \varepsilon_i(s)|\partial_s \tilde{H}(s)|\varepsilon_j(s)\rangle|}{|\varepsilon_i(s) - \varepsilon_j(s)|^2}\ \forall j \neq i \, . \tag{3.3}$$

L'approximation adiabatique est souvent utilisée à partir de l'état fondamental $|\varepsilon_0(t)\rangle$, menant à la définition du gap spectral entre l'état fondamental et le premier état excité du système $\Delta(s) = \varepsilon_1(s) - \varepsilon_0(s)$. Généralement, le maximum de $\langle \varepsilon_i(s)|\partial_s \tilde{H}(s)|\varepsilon_j(s)\rangle$ est de l'ordre d'une valeur propre typique de $\tilde{H}$ et petit. Ainsi, l'équation 3.3 indique que le minimum du carré de l'inverse du gap spectral Δ constitue un critère pratique pour quantifier le temps nécessaire à l'évolution adiabatique.

L'*algorithme adiabatique quantique* (« Quantum Adiabatic Algorithm ») (QAA), introduit par Farhi, Gutmann et Sipser [67], emploie un ordinateur quantique physique pour la résolution de problèmes d'optimisation combinatoire en se basant sur le théorème adiabatique quantique. Pour ce faire, le système physique est initialement préparé dans l'état fondamental d'un hamiltonien de forçage H_D facile à construire et dont l'état fondamental est simple à trouver. La solution du problème, encodée dans l'état fondamental de l'hamiltonien de problème H_P, est alors obtenue en transitionnant de l'état fondamental de l'hamiltonien H_D à l'état fondamental de l'hamiltonien H_P par une évolution adiabatique. Plus précisément, l'hamiltonien du système s'écrit comme

$$\tilde{H}(s) = (1 - s)\, H_D + s H_P\,. \tag{3.4}$$

Ainsi, en présumant que le gap spectral entre l'état fondamental et l'état excité est non nul, la solution du problème est toujours obtenue à partir de l'état fondamental final si l'évolution, donnée par l'opérateur $U(t) = e^{-i\int_0^1 \tilde{H}(s)ds}$, est suffisamment lente comme garanti par le théorème adiabatique quantique. Si le temps d'évolution est trop court, une *transition diabatique*, c'est-à-dire une transition de l'état fondamental à un état excité, peut empêcher l'évolution adiabatique.

Typiquement, le gap spectral est non nul [67], mais cela ne suffit pas à garantir l'efficacité de l'algorithme. En effet, le gap doit suffisamment important pour limiter le temps d'évolution. Pour certains problèmes, cette condition est remplie, permettant une évolution adiabatique en un temps réaliste, mais ce n'est pas toujours le cas [68]. Une alternative consiste à trouver un compromis entre le temps d'évolution et la proximité de l'état final avec l'état fondamental espéré. Le choix du chemin adiabatique utilisé pour transitionner de H_D à H_P peut aussi différer de l'équation 3.4, ce qui peut contribuer à maximiser le gap au long du chemin et donc minimiser le temps d'évolution [69, 70].

L'algorithme adiabatique quantique se place au sein du *calcul adiabatique quantique*, qui regroupe différentes méthodes similaires. Un autre membre de ce groupe, le *recuit quantique*, représente généralement l'emploi de QAA dans un environnement bruité, menant ainsi à une version plus réaliste sans contrainte d'adiabaticité ou d'universalité. Cet algorithme partage plusieurs similitudes avec l'algorithme

quantique d'optimisation approximative qui seront explorées dans les prochaines sections.

3.2 Algorithme quantique d'optimisation approximative

Bien que le calcul adiabatique quantique puisse être utilisé pour résoudre certains problèmes d'optimisation combinatoire, le temps nécessaire à une évolution adiabatique constitue un facteur limitant pour de nombreux problèmes. L'*algorithme quantique d'optimisation approximative* (« Quantum Approximate Optimization Algorithm ») (QAOA) [5] propose alors une alternative, sous la forme d'un algorithme variationnel quantique, discrétisant l'évolution continue de QAA. Cette approche s'éloigne de l'évolution adiabatique en acceptant la présence de transitions diabatiques entre l'état fondamental et les états excités pour réduire le temps d'évolution. L'optimisation des paramètres du circuit quantique paramétré permet de naviguer efficacement l'espace de Hilbert pour atteindre une bonne solution approximative. QAOA est non seulement adiabatique, mais aussi contre-adiabatique, c'est-à-dire qu'il mène à un raccourci à l'adiabacité. En effet, l'erreur de discrétisation peut étonnamment réduire l'impact des excitations diabatiques [71].

3.2.1 Description de l'algorithme

Étant une idée prometteuse pour les applications des ordinateurs quantiques, l'algorithme quantique d'optimisation approximative a mené à une quantité incroyable de travaux dans les précédentes années [72, 73]. Ainsi, pour simplifier la compréhension de ce concept, l'algorithme original, dû à Farhi, Goldstone et Gutmann [5], est d'abord présenté.

QAOA repose sur deux différents hamiltoniens : l'hamiltonien de problème, ou de phase, H_P et l'hamiltonien de forçage, ou de mélange, H_D. L'hamiltonien de problème est formulé de façon à encoder la solution, potentiellement dégénérée, du problème d'optimisation combinatoire dans son état fondamental. Pour ce faire,

celui-ci est défini en fonction de la fonction de coût C de l'instance du problème : $H_P |x\rangle = C(x) |x\rangle$. H_P prend typiquement la forme de l'hamiltonien du modèle d'Ising. L'hamiltonien de forçage, quant à lui, est donné par

$$H_D^X = \sum_{i=1}^{n} X_i \,, \tag{3.5}$$

où X_i est l'opérateur de Pauli X appliqué sur le qubit i d'un système à n qubits. H_D^X est construit de manière à induire de l'interférence et ainsi permettre l'exploration de l'espace de Hilbert.

Par définition, l'hamiltonien H_p est diagonal dans la base computationnelle, alors que l'hamiltonien H_D comprend des termes hors diagonaux de sorte que ceux-ci ne commutent pas entre eux. Deux opérations unitaires paramétrées sont définies à partir des hamiltoniens H_P et H_D : l'opérateur de problème $U_P(\gamma) = e^{-i\gamma H_P}$ ainsi que l'opérateur de forçage $U_D(\beta) = e^{-i\beta H_D}$, où γ et β sont des paramètres réels. L'opérateur U_P représente une rotation de phase, paramétrée par γ, des états de la base computationnelle en fonction de leur énergie donnée par H_P. L'opérateur U_D, paramétré par β, superpose différents états de la base computationnelle ayant précédemment acquis différents facteurs de phase, menant ainsi à de l'interférence.

En tant que VQA, QAOA est un algorithme hybride composé d'un circuit quantique paramétré et d'un optimiseur classique. Le circuit quantique est d'abord préparé dans un état propre de l'hamiltonien de forçage. Pour l'hamiltonien 3.5, un exemple d'état initial possible est la superposition égale des états possibles $= |+\rangle^{\otimes n}$. Le produit des opérateurs $U_D U_P$ est alors appliqué en alternance p fois sur l'état initial $|\psi_0\rangle$, donnant l'état suivant :

$$|\psi(\vec{\gamma}, \vec{\beta})\rangle = \underbrace{U_D(\beta_p)U_P(\gamma_p) \cdots U_D(\beta_1)U_P(\gamma_1)}_{p \text{ fois}} |\psi_0\rangle \,, \tag{3.6}$$

où $\vec{\gamma} = (\gamma_1, \ldots, \gamma_p)$ et $\vec{\beta} = (\beta_1, \ldots, \beta_p)$ sont les paramètres initiaux du circuit. Une fois l'état $|\psi(\vec{\gamma}, \vec{\beta})\rangle$ préparé, la valeur moyenne de l'hamiltonien de problème H_P est calculée par des mesures répétées de l'état final dans la base computationnelle :

$$E_P(\vec{\gamma}, \vec{\beta}) = \langle \psi(\vec{\gamma}, \vec{\beta}) | H_P | \psi(\vec{\gamma}, \vec{\beta}) \rangle \,. \tag{3.7}$$

Comme H_P est typiquement une somme d'opérateurs de Pauli, cette valeur moyenne peut être évaluée efficacement en mesurant l'état du circuit [74]. L'énergie trouvée quantifie l'optimalité de l'état préparé. Par la suite, une méthode d'optimisation classique continue, comme la descente de gradient stochastique, est employée pour mettre à jour itérativement les paramètres $\vec{\gamma}$ et $\vec{\beta}$ du circuit paramétré de manière à minimiser la valeur moyenne $E_P(\vec{\gamma}, \vec{\beta})$:

$$(\vec{\gamma}^*, \vec{\beta}^*) = \arg\min_{\vec{\gamma},\vec{\beta}} E_P(\vec{\gamma}, \vec{\beta})\,. \tag{3.8}$$

Si l'optimisation aboutit de manière espérée, l'état final $|\psi(\vec{\gamma}^*, \vec{\beta}^*)\rangle$ correspond à une superposition des états fondamentaux de H_P et donc aux différentes solutions du problème étudié. Si ce n'est pas le cas, le ratio d'approximation α est utilisé pour décrire la qualité de la solution trouvée comme à la section 1.3 :

$$\alpha(\vec{\gamma}^*, \vec{\beta}^*) = \frac{E_p(\vec{\gamma}^*, \vec{\beta}^*)}{\min_{(\vec{\gamma},\vec{\beta})} E_p(\vec{\gamma}, \vec{\beta})} \tag{3.9}$$

Ce ratio augmente théoriquement avec le nombre de couches p utilisées comme QAOA équivaut à une évolution adiabatique dans la limite où $p \to \infty$ [5]. Notons que pour l'utilisation de QAOA sur des graphes, la valeur de p doit augmenter avec la taille du graphe pour ne pas être limitée par la localité du circuit QAOA [75].

L'algorithme se résume par les étapes suivantes, illustrées à la figure 3.1 :

(1) Définition de l'hamiltonien de problème H_P.
(2) Préparation de l'état initial $|\psi_0\rangle$.
(3) Construction du circuit quantique paramétré $|\psi(\gamma, \beta)\rangle$ en appliquant en alternance les opérateurs $U_P(\gamma)$ et $U_D(\beta)$ p fois.
(4) Calcul de l'énergie E_P à travers de mesures dans la base computationnelle.
(5) Optimisation des paramètres $\vec{\gamma}$ et $\vec{\beta}$ à l'aide d'un optimiseur classique minimisant l'énergie E_P.

Décrivons maintenant plus en détail les mécanismes derrière QAOA : la préparation de l'état initial, l'encodage du problème dans un hamiltonien de problème

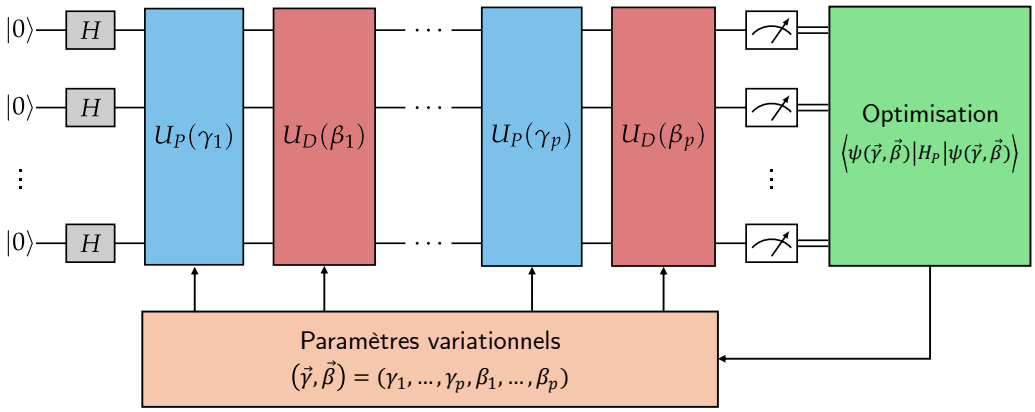

FIGURE 3.1 – Schéma de l'algorithme quantique d'optimisation approximative (QAOA). Un circuit quantique paramétré est préparé en appliquant une alternance d'opérateurs de problème $U_P(\gamma)$ et de forçage $U_D(\beta)$ à un état initial $|+\rangle^{\otimes n}$. La valeur moyenne $\langle\psi(\vec{\gamma},\vec{\beta})|H_P|\psi(\vec{\gamma},\vec{\beta})\rangle$ de l'Hamiltonien de problème H_P est ensuite optimisée en modifiant les paramètres $\vec{\gamma}$ et $\vec{\beta}$ du circuit à l'aide d'un optimiseur classique.

ainsi que le choix de l'hamiltonien de forçage. L'initialisation et l'optimisation des paramètres du circuit sont décrites à la section 3.4.

Préparation de l'état initial

Comme QAA, le circuit quantique est initialement préparé dans l'état fondamental de l'hamiltonien de forçage pour garantir le théorème adiabatique. Cependant, la divergence de QAOA du calcul adiabatique implique qu'un autre état initial peut être choisi. Une alternative consiste à utiliser une solution approximative du problème comme état initial [76]. L'état initial peut aussi être choisi de manière à restreindre l'espace des solutions. Au lieu de l'état initial $|\psi_0\rangle = |0\rangle^{\otimes n}$, un espace ne représentant que les solutions faisables du problème peut être préparé, tel qu'exemplifié à la section 3.3.2.

Encodage du problème

Comment est-ce qu'un problème d'optimisation combinatoire $\varphi(x)$ peut être encodé par un hamiltonien H_P ? D'abord, les entrées x du problème sont caractérisées par une fonction de coût $C(x)$, pénalisant les configurations selon le nombre de

contraintes non respectées. Une entrée optimale est associée à un coût nul, alors qu'une entrée non optimale est associée à un coût positif selon sa proximité avec la solution. Résoudre le problème consiste alors à trouver l'entrée optimale, c'est-à-dire l'entrée x^* minimisant la fonction de coût. L'hamiltonien de problème se définit à partir de la fonction de coût tel que

$$H_P \left|x\right\rangle = C(x) \left|x\right\rangle \ . \tag{3.10}$$

Cette équation implique que l'état fondamental de l'hamiltonien H_P encode les solutions au problème donné. QAOA cherche ainsi à préparer l'état fondamental de H_P comme il correspond à une superposition des solutions au problème φ. Le modèle d'Ising est fréquemment utilisé pour fournir un hamiltonien de problème décrivant la fonction de coût C. En effet, trouver l'état fondamental d'un modèle d'Ising constitue un problème NP-complet, ce qui signifie qu'il existe alors nécessairement une réduction entre ce problème et les différents problèmes NP-complet. Le modèle d'Ising est donné par

$$H_P = -\sum_{(i,j)\in E} J_{ij}\sigma_i\sigma_j - \sum_{i\in V} h_i\sigma_i \,, \tag{3.11}$$

où E est l'ensemble d'arêtes, V est l'ensemble de sommets et σ_i est le spin du site i. Les constantes J et h représentent respectivement l'interaction entre deux sites et le champ magnétique externe. Le problème d'optimisation quadratique binaire non contraint (« Quadratic Unconstrained Binary Optimization ») (QUBO) permet aussi d'exprimer les fonctions de coût de manière plus générale, en ne les restreignant pas au modèle d'Ising.

La formulation de problèmes NP-complet sous la forme du modèle d'Ising n'est pas nécessairement évidente, mais plusieurs réductions ont été formulées pour le calcul adiabatique quantique [77, 78]. Une réduction intéressante dans notre cas relie le problème positif NAE3SAT au modèle d'Ising antiferromagnétique. Une formule CNF φ peut être représentée graphiquement sous la forme d'un graphe biparti nommé *graphe de facteurs*. Dans ce graphe, les clauses et les variables forment les deux types de sommets de la bipartition, alors que les arêtes sont définies entre les sommets de ces deux partitions et correspondent à la présence d'une variable dans une clause. Considérons pour le moment une seule clause $C = (x_1 \vee x_2 \vee x_3)$ de φ

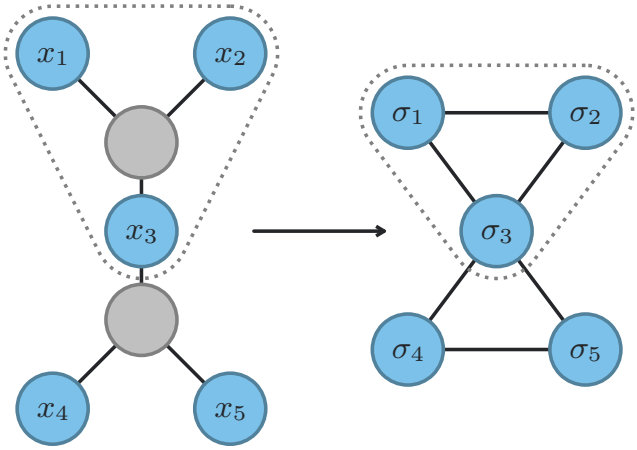

FIGURE 3.2 – Exemple de la transformation entre le modèle d'Ising pour NAE3SAT et 1-in-3SAT pour une formule φ. À gauche, le graphe de facteurs de φ avec les sommets de variables (bleus) et de clauses (gris). À droite, le modèle d'Ising correspondant à φ. Pour 1-in-3SAT, un terme de champ magnétique est ajouté pour favoriser les configurations où chaque clause contient exactement une variable fixée à 1.

et montrons comment transformer celle-ci au modèle d'Ising. En associant chaque variable x_i à un spin σ_i selon la transformation $\sigma_i = 1 - 2x_i$, cette clause se réduit à un modèle d'Ising antiferromagnétique en considérant un réseau triangulaire composé des spins σ_i, où chacun de ceux-ci interagit avec les deux autres spins comme illustré à la figure 3.2. L'hamiltonien d'un tel modèle est donné par

$$H_{P,C}^{\text{NAE3SAT}} = \sigma_1\sigma_2 + \sigma_2\sigma_3 + \sigma_3\sigma_1 \tag{3.12}$$

Pour valider cette transformation, l'énergie associée à chaque combinaison possible de spins est présentée dans le tableau 3.1a. La frustration des spins sur le réseau implique que les seuls états qui ne sont pas dans l'état fondamental sont les états $|\uparrow\uparrow\uparrow\rangle$ et $|\downarrow\downarrow\downarrow\rangle$. Or, il s'agit des deux combinaisons ne faisant pas partie de l'ensemble des solutions du problème positif NAE3SAT. L'état fondamental de l'hamiltonien $H_{P,C}$ correspond ainsi bien aux solutions de la clause C.

Une formule CNF est représentable par un modèle d'Ising en appliquant la transformation précédente sur chacune des clauses C de la formule, tel qu'illustré à

Entrées (spins)	Coût (énergie)
000	3
001	-1
010	-1
100	-1
011	-1
110	-1
101	-1
111	3

(a)

Entrées (spins)	Cout (énergie)
000	0
001	-2
010	-2
100	-2
011	0
110	0
101	0
111	6

(b)

TABLEAU 3.1 – Coût de chaque entrée d'une clause $C = (x_1 \vee x_2 \vee x_3)$ d'une formule CNF pour le problème NAE3SAT (a) et 1-in-3SAT (b). Ce coût correspond à l'énergie du modèle d'Ising donnée par l'hamiltonien $H_{P,C}$ pour une combinaison de spins, reliée aux entrées par la transformation $\sigma_i = 1 - 2x_i$.

la figure 3.2, donnant ainsi l'hamiltonien

$$H_P = \sum_C H_{P,C}\,. \tag{3.13}$$

Cet hamiltonien impose alors les contraintes de chaque clause. Remarquons que l'hamiltonien de problème pour NAE3SAT est donné par l'hamiltonien d'Ising donné à l'équation 3.11 avec $J_{ij} = -1$ et $h_i = 0$ pour un graphe construit comme à la figure 3.2. Similairement, le problème 1-in-3SAT se transforme au modèle d'Ising en ajoutant pour chaque clause l'hamiltonien

$$H_{P,C}^{\text{1-in-3SAT}} = \sigma_1\sigma_2 + \sigma_2\sigma_3 + \sigma_3\sigma_1 - \sigma_1 - \sigma_2 - \sigma_3\,. \tag{3.14}$$

Un champ magnétique externe est ajouté sur chaque spin pour imposer la contrainte supplémentaire, c'est-à-dire que toutes les clauses doivent contenir exactement une variable évaluant à un vrai logique. Le tableau 3.1b présente les énergies des configurations pour l'hamiltonien précédent, validant la correspondance entre les états fondamentaux et les solutions du problème SAT. L'hamiltonien pour le problème 1-in-3SAT est ainsi donné par l'hamiltonien d'Ising donné à l'équation 3.11 avec $J_{ij} = -1$ et $h_i = 1$.

Choix du forçage

L'hamiltonien de forçage H_D initialement proposé avec QAOA prend son inspiration de l'algorithme adiabatique quantique, en utilisant l'hamiltonien facile à préparer H_D^X. Cela implique que toute la dépendance au problème doit être encodée dans l'hamiltonien de problème H_P. Comme QAOA n'est pas restreint par cette condition, des opportunités se présentent pour encoder différemment le problème. Par exemple, l'hamiltonien de forçage peut être utilisé pour restreindre l'espace de Hilbert en prenant en compte la structure du problème. Divers hamiltoniens offrent différentes performances selon le problème étudié. Le meilleur choix de forçage demeure toutefois une question ouverte.

3.2.2 Relation avec l'algorithme adiabatique quantique

QAA requiert une évolution continue de l'état, alors que QAOA repose sur l'application de portes quantiques. Pour établir un lien entre QAA et QAOA, l'évolution de l'hamiltonien de QAA doit donc être rendue discrète. Pour ce faire, l'évolution unitaire de QAA est émulée avec des portes quantiques en décomposant l'opérateur d'évolution $U(t) = e^{-i\int_0^T H(t)dt}$ de l'hamiltonien de QAA en séquence de petits pas de temps [73], tel que

$$U(t) \approx \prod_{k=0}^{t/\Delta t-1} e^{-iH(k\Delta t)\Delta t}\,, \tag{3.15}$$

pour un intervalle de temps Δt. Par la suite, la décomposition de Suzuki-Trotter, donnée par $e^{(A+B)t} = \lim_{n\to\infty}(e^{At/n}e^{Bt/n})^n$ pour deux opérateurs A et B, est employée. Le premier ordre de cette décomposition permet de discrétiser l'opérateur de l'hamiltonien dépendant du temps $H(t) = (1-\frac{t}{T})H_D + \frac{t}{T}H_P$ de QAA (voir l'équation 3.4), où $t \in [0, T]$, tel que

$$e^{-iH(t)} \approx e^{-i(1-\frac{t}{T})H_D\Delta t}e^{-i\frac{t}{T}H_P\Delta t} + O(\Delta t^2)\,. \tag{3.16}$$

La forme de l'opérateur d'évolution de QAOA pour une profondeur $p = 1$ de circuit est retrouvée en prenant $\beta = (1-\frac{t}{T})\Delta t$ et $\gamma = (\frac{t}{T})\Delta t$ [79]. Combinant cette dernière

équation avec l'équation 3.15, l'opérateur d'évolution devient

$$U(t) \approx \prod_{k=0}^{t/\Delta t-1} e^{-i(1-\frac{k\Delta t}{T})H_D\Delta t} e^{-i\frac{k\Delta t}{T}H_P\Delta t}. \tag{3.17}$$

Ce processus est généralement nommé *évolution adiabatique trottérisée* en référence à la décomposition de Suzuki-Trotter. Dans la limite $p \to \infty$, une évolution adiabatique est retrouvée, car la condition adiabatique est satisfaite. Pour un circuit de profondeur p finie, plus les paramètres γ et β sont faibles, plus les erreurs découlant de la discrétisation, communément appelées *erreurs de Trotter*, sont atténuées, car le pas de temps d'évolution est raccourci. Cependant, cela diminue aussi la durée totale d'évolution adiabatique T, augmentant ainsi l'impact des excitations diabatiques qui nuisent à la performance de l'algorithme. Au contraire, des paramètres plus grands allongent le temps d'évolution aux dépens de l'augmentation des erreurs de Trotter. Un compromis entre ces deux facteurs doit être choisi pour minimiser les erreurs.

3.3 Ansatz quantique à opérateurs alternants

L'algorithme quantique d'optimisation approximative applique en alternance des opérateurs basés sur l'hamiltonien de problème et l'hamiltonien de forçage, guidé par l'algorithme quantique adiabatique. Cet algorithme comporte une limitation importante : les opérateurs appliqués doivent prendre la forme d'une évolution temporelle d'un hamiltonien local fixe. Cette restriction entrave la construction d'opérateurs unitaires potentiellement plus efficaces.

L'*ansatz quantique à opérateurs alternants* (« Quantum Alternating Operator Ansatz») (QAOA), introduit par Hadfield et coll. [4], généralise l'algorithme quantique d'optimisation approximative en permettant l'alternance de familles générales d'opérateurs unitaires paramétrés plutôt qu'uniquement des opérateurs basés sur un hamiltonien. Notons qu'un *ansatz* décrit typiquement une sous-routine composée d'une séquence de portes appliquées sur des qubits spécifiques. Cette approche supporte ainsi la représentation d'un plus grand nombre d'états, pouvant possiblement

être construite de manière plus efficace. De plus, celle-ci facilite l'utilisation d'opérateurs plus facilement réalisables sur le matériel informatique quantique actuel. Cette extension trouve toutefois son utilité principale dans la création d'opérateurs de forçage. En effet, ces opérateurs peuvent désormais être construit de manière à restreindre l'espace des configurations d'un problème selon ses contraintes et ainsi éviter une recherche de l'espace de Hilbert complet.

L'algorithme quantique d'optimisation approximative et l'ansatz quantique à opérateurs alternants possèdent le même acronyme. Comme cette dernière étend le premier, l'acronyme QAOA fera référence à l'ansatz quantique à opérateurs alternants pour le reste de la présente section.

3.3.1 Description de l'approche

Soit un problème d'optimisation combinatoire $\varphi(x)$ et une fonction de coût $C(x)$ décrivant la qualité des entrées x de φ. Généralisant l'algorithme de Farhi, Goldstone et Gutmann, QAOA est constitué de deux familles d'opérateurs : les opérateurs de séparation de phase $U_P(\gamma)$ et les opérateurs de forçage $U_D(\beta)$, où γ et β sont des paramètres réels. Notons que U_P et U_D ne sont pas restreints à la classe d'opérateurs $e^{-i\theta H}$. Le circuit QAOA est construit en appliquant en alternance p couches d'opérateurs des deux familles précédentes à un état initial donné.

Plutôt que de prendre en compte l'espace de Hilbert complet, un *sous-espace faisable F* est considéré. Cet espace ne représente qu'un sous-ensemble de l'espace des configurations possibles, où seules les configurations respectant les contraintes du problème sont présentes. La restriction provient de la famille d'opérateurs de forçage encodant les différentes contraintes du problème φ. En général, l'état initial est construit à partir d'une solution faisable alors que l'opérateur de forçage s'assure de restreindre les états possibles au sous-espace faisable. QAOA est donc particulièrement utile pour les problèmes comportant des contraintes rigides. Par exemple, le problème de recouvrement de sommets maximal nécessite que les solutions prennent la forme d'un état de Dicke, c'est-à-dire une superposition de tous les états de même poids d'Hamming. Intuitivement, limiter l'espace des configurations devrait améliorer la performance de l'algorithme.

Quelques restrictions sont appliquées dans la conception des composantes de QAOA. D'abord, l'état initial doit être trivial à réaliser, c'est-à-dire réalisable avec un circuit de profondeur constante. Les unitaires de séparation de phases doivent être diagonales dans la base computationnelle et sont prises dans la majorité des cas comme $U_P(\gamma) = e^{-i\gamma H_P}$. La construction des unitaires de forçage est ici d'un plus grand intérêt. Ceux-ci doivent préserver le sous-espace faisable et donc transformer les états faisables en états faisables. Les unitaires de forçage doivent aussi fournir des transitions entre toutes les paires d'états correspondant aux états faisables.

3.3.2 Forçage de Grover

L'ansatz quantique à opérateurs alternants avec forçage de Grover (« Grover-Mixer Quantum Alternating Operator Ansatz ») (GM-QAOA) fut proposé par Bärtschi et Eidenbenz [6] pour déplacer la complexité de la conception de l'opérateur de forçage à la préparation de l'état initial en s'inspirant de l'algorithme de Grover. Cet algorithme se base sur le cadre théorique défini dans la section précédente pour produire efficacement une superposition égale de toutes les solutions faisables.

Décrivons d'abord rapidement le circuit GM-QAOA, tel qu'illustré à la figure 3.3. Un opérateur unitaire de préparation d'état U_S crée une superposition égale de toutes les solutions réalistes $|F\rangle$ dans le sous-espace faisable F. Par la suite, similairement à QAOA, les opérateurs unitaires de problème U_P et de forçage U_M sont appliqués en alternance p fois.

Plutôt que de préparer un état initial à partir d'une seule solution faisable, comme QAOA, l'état initial de GM-QAOA est préparé dans une superposition de l'entièreté des solutions. Cette distinction entraine une perspective différente de celle adoptée par QAOA. La difficulté provient désormais de la préparation de l'état plutôt que de la conception de l'opérateur de mélange. L'opérateur de mélange U_D quant à lui, est donné par

$$U_D^{\text{Grover}} = e^{-i\beta|F\rangle\langle F|} = U_S \left[\mathbb{1}^{\otimes n} - \left(1 - e^{-i\beta}\right)(|0\rangle\langle 0|)^{\otimes n}\right] U_S^\dagger . \tag{3.18}$$

Cet opérateur s'apparente aux opérateurs de diffusion utilisés dans l'algorithme d'amplification d'amplitude, où les opérateurs de déphasage $e^{-i\beta}$ remplacent les

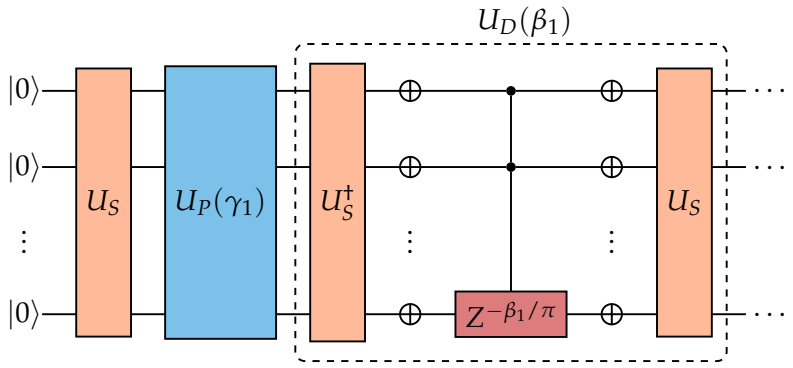

FIGURE 3.3 – Première couche du circuit de l'ansatz quantique à opérateurs alternants (GM-QAOA). Ce circuit est composé de l'opérateur de préparation d'état U_S, de l'opérateur de problème U_P, de portes de Pauli X ainsi que d'une porte de déphasage $Z^{-\beta\pi}$ contrôlée sur tous les qubits. L'opérateur de forçage U_D regroupe les opérateurs à l'intérieur de la boite pointillée.

opérateurs d'inversion de phase $e^{-i\pi}$. U_D est alors réalisé en utilisant les unitaires U_S et $U_S^\dagger$, deux couches d'opérateur de Pauli X et un opérateur de déphasage $Z^{-\beta/\pi} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\beta} \end{pmatrix}$ contrôlés sur tous les qubits.

L'influence de l'algorithme de Grover sur GM-QAOA a notamment pour conséquence que GM-QAOA génère des superpositions uniformes d'états ayant la même énergie. Cette propriété est d'un intérêt particulier pour ce projet, comme elle garantit la distribution de solutions uniforme nécessaire à l'algorithme de JVV. De plus, comme cette approche est réalisable directement à partir d'un ensemble de portes quantiques standard, aucune erreur de simulation d'hamiltonien, comme les erreurs de Trotter, affecte l'algorithme. Notons que les problèmes NAE3SAT et 1-in-3SAT ne sont pas contraints et donc que l'espace faisable demeure le même que pour QAOA, c'est-à-dire que $|F\rangle = |+\rangle^{\otimes n}$.

3.4 Configuration des paramètres

La capacité d'entraînement des réseaux de neurones à partir d'une simple descente de gradient fut en partie à l'origine de leur succès retentissant sur de nombreux différents problèmes. L'optimisation des paramètres se fait simplement, même avec des fonctions de coût non convexes, tout en offrant de puissants résultats. Une des attentes envers les VQA était qu'ils possèdent ce même comportement et qu'ils puissent ainsi repousser les limites des algorithmes variationnels. En effet, l'optimisation classique des paramètres de QAOA fait partie intégrante du concept d'algorithme variationnel quantique. Grâce à celle-ci, il est en théorie possible de tirer avantage des algorithmes quantiques en recourant à des ordinateurs quantiques bruités. Cependant, de nombreuses embûches rendent actuellement cette tâche difficile.

D'abord, l'estimation du gradient de coût est difficile dans de nombreuses situations en raison de la présence de plateaux stériles [80, 81], où le gradient devient exponentiellement petit avec la taille du système. Un nombre exponentiel de mesures devient alors nécessaire pour pouvoir identifier la direction minimisant la fonction de coût. Ces complications se présentent surtout pour des circuits de grande profondeur, mais d'autres problèmes surgissent même pour ceux de petites tailles. Un grand nombre de minima locaux sont effectivement considérés comme pauvres, c'est-à-dire qu'ils possèdent une énergie petite par rapport au minimum global, augmentant la difficulté d'atteindre une solution approximative de bonne qualité [82]. De plus, l'optimisation des paramètres des VQA est NP-difficile, et donc intraitable dans le pire des cas [83]. Ces difficultés sont d'autant plus importantes comme la complexité de l'espace des paramètres augmente avec le nombre de paramètres d'un circuit, ce qui implique une plus grande difficulté d'optimisation pour les circuits profonds.

Cet ensemble de complications implique alors qu'une bonne initialisation des paramètres est nécessaire au succès de QAOA. Des paramètres initiaux suffisamment près de l'extremum global peuvent aider à contourner les problèmes énoncés précédemment. Le choix de bons paramètres demeure toutefois une question ouverte. Parmi les différentes approches, Sack et Serbyn proposent une stratégie d'initiali-

sation basée sur le *recuit quantique trottérisé* (« Trotterized Quantum Annealing ») (TQA) [79]. Cette méthode, utilisée pour les simulations de ce travail, offre la même performance qu'un nombre exponentiel d'initialisations aléatoires. Comme vu à la section 3.2.2, un juste milieu doit être choisi entre le temps d'évolution et les erreurs de Trotter. L'initialisation TQA permet de trouver le temps d'évolution optimal à une profondeur de circuit p fixe. Pour ce faire, la procédure décrite à la section 3.2.2 est appliquée sur une grille uniformément discrétisée des temps d'évolution $t_k = k\Delta t$, avec $k = 1, \ldots, p$, pour un pas de temps $\Delta t = T/p$. Les angles du circuit QAOA correspondant sont alors :

$$\gamma_i = \frac{i}{p}\Delta t\,, \beta = (1 - \frac{i}{p})\Delta t\,. \tag{3.19}$$

Le pas de temps est optimisé classiquement en mesurant la valeur moyenne de l'hamiltonien de problème H_P de manière à minimiser cette dernière, donnant ainsi de bons paramètres initiaux.

3.5 Échantillonnage et biais

L'allure de la distribution obtenue par QAOA est dans notre cas essentiel. Afin de se conformer à la condition de l'algorithme de JVV, celle-ci doit être quasi uniforme, c'est-à-dire qu'elle doit posséder une faible non-uniformité. Est-ce le cas ? D'abord, à moins que l'état préparé soit optimal, des non-solutions sont nécessairement présentes dans la distribution obtenue, bien que potentiellement avec une faible probabilité. Comme l'algorithme de JVV nécessite une distribution constituée uniquement de solutions, cela pose problème. Une étape de post-sélection est alors nécessaire pour retirer les non-solutions des échantillons mesurés. Heureusement, vérifier la validité d'une entrée se fait efficacement pour les problèmes NP. Toutefois, rien ne nous garantit que la distribution restante, composée uniquement de solutions, est effectivement non-uniforme.

En général, le recuit quantique n'échantillonne pas les états fondamentaux uniformément [84, 85]. Certains états sont exponentiellement supprimés et nécessitent un nombre exponentiel de mesures pour être détectés. Comme le recuit quantique et

QAOA sont fortement liés, il est ainsi peu probable que QAOA puisse échantillonner les états fondamentaux de manière uniforme.

Une propriété de l'algorithme GM-QAOA devient alors d'un intérêt considérable : l'équiprobabilité des états de même énergie. Les solutions étant encodées dans l'état fondamental de l'hamiltonien de problème, elles possèdent ainsi la même amplitude et donc la même probabilité. En ne considérant que les solutions, l'état préparé par le circuit GM-QAOA possède toujours une non-uniformité nulle, un avantage non négligeable par rapport à QAOA.

Chapitre 4

Comptage variationnel quantique

Les chapitres précédents ont laissé transparaître l'intuition derrière l'approche empruntée dans ce travail : est-il possible d'utiliser les algorithmes variationnels quantiques comme générateur de solutions à l'algorithme de JVV pour la résolution approximative des problèmes de comptage ? Avant d'introduire le fruit du travail de ce mémoire, l'algorithme VQCount, énumérons les difficultés liées à la résolution de problèmes de comptage avec les algorithmes variationnels quantiques. D'abord, les circuits quantiques doivent produire une grande séparation entre les amplitudes des solutions et les non-solutions de l'état produit pour pouvoir produire même une seule solution avec un nombre tractable de mesures. Ensuite, même si les amplitudes des non-solutions sont nulles, si les amplitudes d'un sous-ensemble non négligeable des solutions sont significativement supprimées par rapport aux autres amplitudes, la mesure des états supprimés entraîne une surcharge computationnelle, potentiellement exponentielle, du nombre de répétitions. Finalement, même si toutes les solutions possèdent environ des amplitudes égales et si les non-solutions ne sont pas présentes dans l'état produit, un nombre exponentiel de mesures est en théorie nécessaire pour énumérer naïvement les solutions de manière exhaustive, comme le nombre de solutions est exponentiel pour les problèmes d'intérêt.

L'apport principal de ce travail réside dans la combinaison des algorithmes variationnels quantique et l'algorithme de JVV. L'algorithme VQCount échantillonne la distribution préparée par GM-QAOA afin de produire un compte approximatif à l'aide de l'algorithme de JVV. Cette approche tente de minimiser l'impact des obs-

tacles précédents, en particulier de la deuxième et troisième difficulté. L'utilisation de GM-QAOA permet d'éviter la suppression d'amplitude d'un sous-ensemble de solutions en assurant que toutes les solutions aient la même amplitude. De plus, l'algorithme de JVV tire avantage de la structure des problèmes auto-réductibles pour éviter de devoir énumérer naïvement toutes les solutions. Afin de prendre en compte la propriété d'auto-réductibilité, le circuit GM-QAOA doit nécessairement être modifié. Une procédure d'auto-réduction, comprise dans l'algorithme VQCount, modifie ainsi itérativement le circuit pour la résolution des sous-problèmes tout en conservant les propriétés de GM-QAOA.

Ce chapitre débute en introduisant l'algorithme VQCount à la section 4.1 et décrit la procédure d'auto-réduction nécessaire à l'algorithme VQCount à la section 4.2.

4.1 Algorithme VQCount

Comment faire le pont entre les algorithmes variationnels quantiques et l'algorithme de JVV pour la résolution de problème de comptage ? A priori, l'algorithme de JVV doit tout simplement être implémenté en utilisant un algorithme variationnel quantique comme générateur de solutions. Cependant, plusieurs embûches barrent notre chemin. Décrivons la procédure, nommée VQCount par brièveté, en affrontant ces problèmes en chemin.

Soit une instance de problème SAT, décrite par la formule CNF φ. Ce problème est auto-réductible par la relation 2.1. Ce travail se limite à l'étude du problème SAT en gardant en tête que, par sa NP-complétude, une réduction existe entre n'importe quel problème de la classe NP et celui-ci. La formule φ peut être transformée en un modèle d'Ising, tel que décrit à la section 3.2.1, de sorte que l'hamiltonien du modèle d'Ising H_P encode le problème. Grâce à cet hamiltonien de problème, le circuit paramétré quantique de l'algorithme QAOA est construit en préparant l'état $|\psi(\vec{\gamma}_0, \vec{\beta}_0)\rangle$ avec les paramètres initiaux $\vec{\gamma}_0$ et $\vec{\beta}_0$. Négligeons pour le moment le choix de l'état initial et de l'hamiltonien de forçage. Une fois le circuit construit, l'énergie moyenne de H_P est évaluée à l'aide de mesures répétées de l'état préparé. Un optimiseur classique modifie ensuite les paramètres du circuit quantique pour minimiser cette énergie, donnant un circuit quantique paramétré optimisé $|\psi(\vec{\gamma}, \vec{\beta})\rangle$.

Les échantillons, obtenus par des mesures de l'état préparé par le circuit dans la base computationnelle, contiennent alors des solutions à la formule φ avec une haute probabilité.

QAOA étant une méthode heuristique, les paramètres optimaux ne sont pas nécessairement atteints, impliquant la présence de non-solutions dans la distribution obtenue en mesurant $|\psi(\vec{\gamma},\vec{\beta})\rangle$. L'algorithme de JVV nécessite pourtant une distribution composée uniquement de solutions, impliquant qu'une étape de post-traitement doit être employée pour retirer les non-solutions. Comme le problème SAT appartient à la classe NP, vérifier qu'un échantillon est une solution est un processus efficace. Ainsi, toutes les non-solutions échantillonnées sont retirées de la distribution obtenue. De plus, comme discuté à la section 3.5, la non-uniformité de la distribution obtenue ne respecte pas nécessairement la condition de l'algorithme de JVV. La discussion dans cette section se restreint alors à la variante GM-QAOA, qui garantit que les amplitudes des solutions préparées soient égales. Ainsi, le critère de mérite pertinent est la probabilité que le résultat d'une mesure soit une solution, ou plus succinctement le *taux de succès*

$$r = \langle\psi(\vec{\gamma},\vec{\beta})|\hat{\mathcal{P}}_G|\psi(\vec{\gamma},\vec{\beta})\rangle \ , \tag{4.1}$$

où $\hat{\mathcal{P}}_G$ est le projecteur dans l'espace des solutions G, c'est-à-dire l'état fondamental du modèle d'Ising représentant le problème, donné par

$$\hat{\mathcal{P}}_G = \sum_{x\in G} |x\rangle\langle x| \ . \tag{4.2}$$

Cette définition permet alors de décrire la performance des différents algorithmes pour le comptage basé sur l'échantillonnage sur un pied d'égalité en utilisant le taux de succès r, la tolérance (ou l'erreur multiplicative) ε ainsi que la confiance δ. Cette caractérisation s'étend aux algorithmes classiques employant un générateur échouant à produire une solution à un taux $1-r$.

Ces remarques complétées, des échantillons sont mesurés à partir du circuit $|\vec{\gamma},\vec{\beta}\rangle$ pour estimer la valeur la plus probable w_1 de la première variable x_1. Selon l'algorithme de JVV, il faut alors échantillonner des solutions à la sous-instance φ_{w_1}. Résoudre directement celle-ci à l'aide du même circuit optimisé demanderait une

post-sélection supplémentaire sur la première variable. En réalité, comme il existe un nombre exponentiel de sous-instances en raison du nombre exponentiel de chaînes de bits possibles, un nombre exponentiel d'échantillons supplémentaires devraient être échantillonnés. Une autre méthode consiste à construire et optimiser un différent circuit pour résoudre la sous-instance. Pour éviter cette surcharge computationnelle, le circuit de GM-QAOA est modifié pour résoudre directement la sous-instance. Cette procédure, nommée procédure d'auto-réduction, consiste à remplacer, pour chaque qubit fixé, la porte d'Hadamard par une porte de Pauli conditionnée sur la valeur w_0 du bit fixé et à retirer la composante de l'opérateur de forçage sur le qubit associé. Cette procédure est détaillée à la section 4.2. Ce changement permet de conserver la propriété d'égalité des amplitudes de GM-QAOA. De plus, celle-ci suggère qu'il soit possible d'éviter d'optimiser à nouveau le circuit en gardant les mêmes paramètres. Ainsi, le circuit est modifié selon la procédure d'auto-réduction sans procéder à une nouvelle ronde d'optimisation. Des solutions sont alors encore échantillonnées de ce sous-circuit et le processus est répété de manière récursive jusqu'à ce que toutes les variables soient fixées. Le nombre de solutions est alors donné par l'inverse du produit des probabilités conditionnelles tel qu'énoncé par l'algorithme de JVV. L'algorithme 1 résume les étapes de l'algorithme VQCount.

Comme l'algorithme VQCount se base sur celui de JVV, son utilisation requiert $O(\frac{n^2 \log(\delta^{-1})}{r\varepsilon^2})$ échantillons pour estimer le nombre d'états fondamentaux à une erreur multiplicative ε avec une confiance $1-\delta$ pour un taux de succès r. Ce résultat se compare favorablement à un travail similaire évaluant la fonction de partition d'hamiltoniens de spins classiques qui nécessite $O(\frac{\sqrt{N \log(\delta^{-1})}}{r\varepsilon})$ échantillons pour la même tâche [86]. Lorsque $N = O(2^n)$, VQCount nécessite exponentiellement moins d'échantillons que cette méthode. Notons qu'un algorithme classique peut être utilisé pour résoudre l'instance du problème de comptage lorsque VQCount a suffisamment réduit celle-ci.

Bien que l'approche présentée règle certains des problèmes énoncés au début de la section, les problèmes #P-difficile demeurent difficiles pour les ordinateurs quantiques. La génération des distributions arbitraires, incluant les distributions uniformes de structures combinatoires, est difficile en général et seulement facile dans des cas exceptionnels [14, 15]. De plus, GM-QAOA requiert en général des profondeurs de circuits de taille exponentielle selon la taille du problème pour

Algorithme 2 : VQCount

Entrée: Nombre de variables : n, Hamiltonien de problème : H_P, Hamiltonien de forçage : H_D, Paramètres initiaux : $(\vec{\gamma}_0, \vec{\beta}_0)$, Profondeur : p, Nombre d'étapes d'optimisation : n_o, Nombre de solutions à échantillonner : n_s

PQC$(\vec{\gamma}_0, \vec{\beta}_0) \leftarrow$ Circuit-QAOA$(H_P, H_D, \vec{\gamma}_0, \vec{\beta}_0, p)$
PQC$(\vec{\gamma}, \vec{\beta}) \leftarrow$ Optimisation(PQC$(\vec{\gamma}_0, \vec{\beta}_0), n_o)$
$\tilde{N} \leftarrow 1$
pour $i \in \{1, \ldots, n\}$ **faire**
  $S \leftarrow \{\ \}$
  tant que $|S| < n_s$ **faire**
    $m \leftarrow$ Mesure(PQC$(\vec{\gamma}, \vec{\beta})$)
    si Vérification-Solution(m, H_P) **alors**
      $S \leftarrow S \cup \{m\}$
    fin si
  fin tant que
  $w, \tilde{p} \leftarrow$ Préfixe-Majoritaire(S)
  PQC$(\vec{\gamma}, \vec{\beta}) \leftarrow$ Auto-Réduction(PQC$(\vec{\gamma}, \vec{\beta}), w)$
  $\tilde{N} \leftarrow \tilde{N}/\tilde{p}$
fin pour
retourne $\tilde{N}$

atteindre un taux de succès fini [87].

4.2 Procédure d'auto-réduction

Pour agir en tant que générateur de solutions à la sous-instance de l'instance du problème original, le circuit GM-QAOA doit être modifié pour être en mesure d'échantillonner à partir de la distribution de solutions appropriée. Comme expliqué à la section précédente, il n'est pas suffisant d'échantillonner l'état produit par le circuit original en raison du nombre exponentiel de mesures supplémentaires nécessaires. Une *procédure d'auto-réduction*, illustrée à la figure 4.1, est alors introduite pour résoudre cette complication. Soit une formule CNF φ de taille n et une sous-formule CNF φ_w représentant la formule φ où les premières variables sont remplacées par w. Alors, pour chaque qubit q fixé à w_q du circuit GM-QAOA, les opérations suivantes sont appliquées :

(1) Retirer la porte d'Hadamard H initiale du qubit q.
(2) Insérer un porte de Pauli X classiquement conditionnée par la valeur w_q.
(3) Retirer l'opérateur de l'hamiltonien de forçage H_D du qubit q.

Ces modifications fixent alors l'état du qubit q à $|w_q\rangle$ tout en préservant les termes d'interactions encodant les contraintes de la formule φ_w. Comme l'opérateur de forçage U_D n'est plus appliqué au qubit q, celui-ci demeure dans le même état, à une phase près, spécifié par la porte de Pauli X. En effet, l'opérateur de problème n'applique qu'une phase globale sur le qubit q. Le sous-circuit, c'est-à-dire le circuit initial sans le qubit q, résout alors le sous-problème tout en conservant les mêmes propriétés de GM-QAOA. En effet, similairement à propriété d'auto-réductibilité du problème SAT, le modèle d'Ising est aussi auto-réductible. Considérons un terme d'interaction Z_qZ_p de ce modèle appliqué sur les spins q et p. Fixer le spin q à une valeur constante σ_q signifie que l'influence de ce terme sur le spin p peut être vu comme un terme de champ magnétique local σ_qZ_p. Le sous-circuit peut ainsi être reconstruit en tirant avantage de la propriété d'auto-réductibilité du modèle d'Ising. Avec cette perspective, nous remarquons que le sous-circuit constitue aussi un circuit GM-QAOA.

Le développement de cette procédure a aussi mené à une expression récursive fermée de l'état préparé par le circuit GM-QAOA, présenté dans l'annexe A.

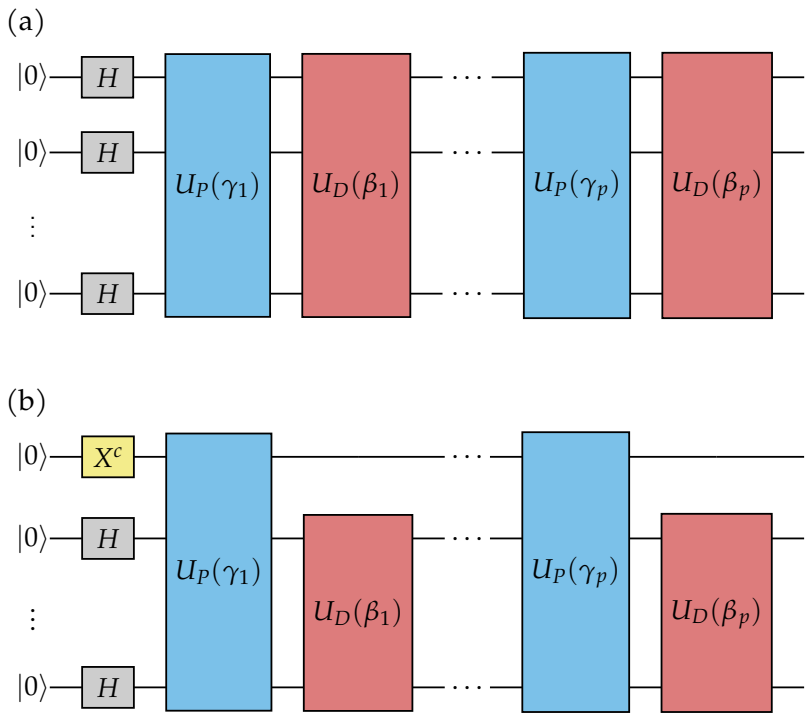

FIGURE 4.1 – Générateur de solutions QAOA avant (a) et après (b) avoir fixé le premier qubit à c durant la procédure d'auto-réduction. La porte d'Hadamard H agissant sur le qubit fixé est remplacée par la porte de Pauli-X, conditionnée classiquement sur c. Pour chaque couche, l'opérateur de problème $U_P(\gamma)$ reste inchangé, tandis que les portes de l'opérateur de forçage $U_D(\beta)$ agissant sur le qubit fixé sont retirées.

L'algorithme VQCount ne comprend qu'une optimisation des paramètres du circuit quantique paramétré avant la procédure d'auto-réduction. Toutefois, un autre cycle d'optimisation est en principe nécessaire pour chaque circuit réduit afin d'atteindre un état optimal représentant les solutions à la sous-instance du problème. Ce choix mène alors à la question suivante : est-ce que la probabilité de succès r du circuit réduit est égale ou supérieure à celle du circuit original avec les mêmes paramètres ? Dans ce cas, l'optimisation n'est nécessaire que pour le circuit original et peut être évitée pour les circuits réduits.

La réponse à cette question est positive dans le cas de circuits GM-QAOA peu profonds dans la limite $\gamma_i = \beta_i$, $i = 1, \ldots, p$. Dans cette limite, GM-QAOA se réduit à l'algorithme de Grover. En effet, comme décrit ci-dessous, pour tous les hamiltoniens de problème H_P pouvant être écrit comme une somme de chaînes de

Pauli, un circuit peut être construit pour effondrer tous les niveaux excités de H_P à un seul niveau excité, ce qui donne un hamiltonien d'évolution à deux niveaux H_P. Avec les paramètres susmentionnés, l'opérateur de problème U_P devient un oracle pour les états fondamentaux de H_P, à une phase constante près, et U_D devient l'opérateur de diffusion de Grover. Dans cette limite, à une profondeur constante $p = O(1)$, le taux de succès est une fonction strictement croissante du rapport des solutions au nombre total d'états. Si, en descendant l'arbre d'auto-réductibilité de l'algorithme de JVV, nous choisissons les branches possédant une probabilité plus grande que $1/2$, alors le ratio des solutions au nombre total d'états est aussi strictement croissant. Cela signifie que, pour ce choix de paramètres, un circuit GM-QAOA peu profond pour un problème donné peut être modifié selon la procédure d'auto-réduction pour résoudre les sous-problèmes avec le même ou un meilleur taux de succès. Cet argument est seulement valide dans la limite de Grover, mais suggère que la stratégie d'auto-réduction est possiblement aussi valide dans d'autres cas. Le chapitre suivant introduit des évidences sous la forme de simulations numériques pour confirmer la validité de cette intuition.

Comment est-il possible d'effondrer une multitude de niveaux excités à un unique niveau excité pour un hamiltonien H_P à l'aide d'un circuit ? Montrons un exemple avec le circuit illustré à la figure 4.2. Ce circuit effondre les états excités pour un problème NAE3SAT décrit par la formule CNF $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$. Le premier registre du circuit, composé de trois qubits d'entrées, deux qubits de parité et un qubit de contrôle, concerne la première clause. De même, un deuxième registre, identique au premier registre, concerne la deuxième clause. Le dernier qubit de phase s'occupe de l'application d'une phase aux solutions.

Pour comprendre ce circuit, déterminons d'abord l'état des qubits du premier registre juste avant l'application de l'opérateur de phase multicontrôlé $Z^{-\gamma\pi}$. Les qubits d'entrées correspondent aux valeurs prises par les littéraux de la première clause de φ. La parité des couples de littéraux (x_1, x_2) et (x_2, x_3) de cette clause est d'abord sauvegardée dans les deux qubits de parité à l'aide de portes CNOT. Par la suite, une porte NOT est appliquée sur les qubits de parité. Finalement, une porte de Toffoli transforme le qubit de contrôle si l'état des qubits de parité est $|11\rangle$. Le tableau 4.1 présente les valeurs des qubits mentionnés selon les différentes valeurs des qubits d'entrée à ce point du circuit.

Entrée	Parité	Contrôle
000	11	1
001	10	0
010	00	0
100	01	0
011	01	0
101	00	0
110	10	0
111	11	1

TABLEAU 4.1 – État des différents qubits du premier registre du circuit d'effondrement des états excités avant l'application de l'opérateur de phase $Z^{-\gamma\varepsilon\pi}$.

Remarquons que le qubit de contrôle est dans l'état $|1\rangle$ seulement si les qubits d'entrée sont dans l'état $|000\rangle$ ou $|111\rangle$, c'est-à-dire les non-solutions du problème NAE3SAT. Le qubit de contrôle indique donc si une clause est satisfaite ou insatisfaite. Cette procédure s'applique aussi au deuxième registre, où le qubit de contrôle de celui-ci indique si la deuxième clause est respectée. Sachant cela, un opérateur de phase $Z^{-\gamma\varepsilon\pi}$ anti-contrôlé sur tous les qubits de contrôle permet d'appliquer une phase $e^{-i\gamma\varepsilon}$ uniquement aux solutions. Après l'application de cette phase, l'inverse du circuit est appliqué pour rétablir l'état initial [74].

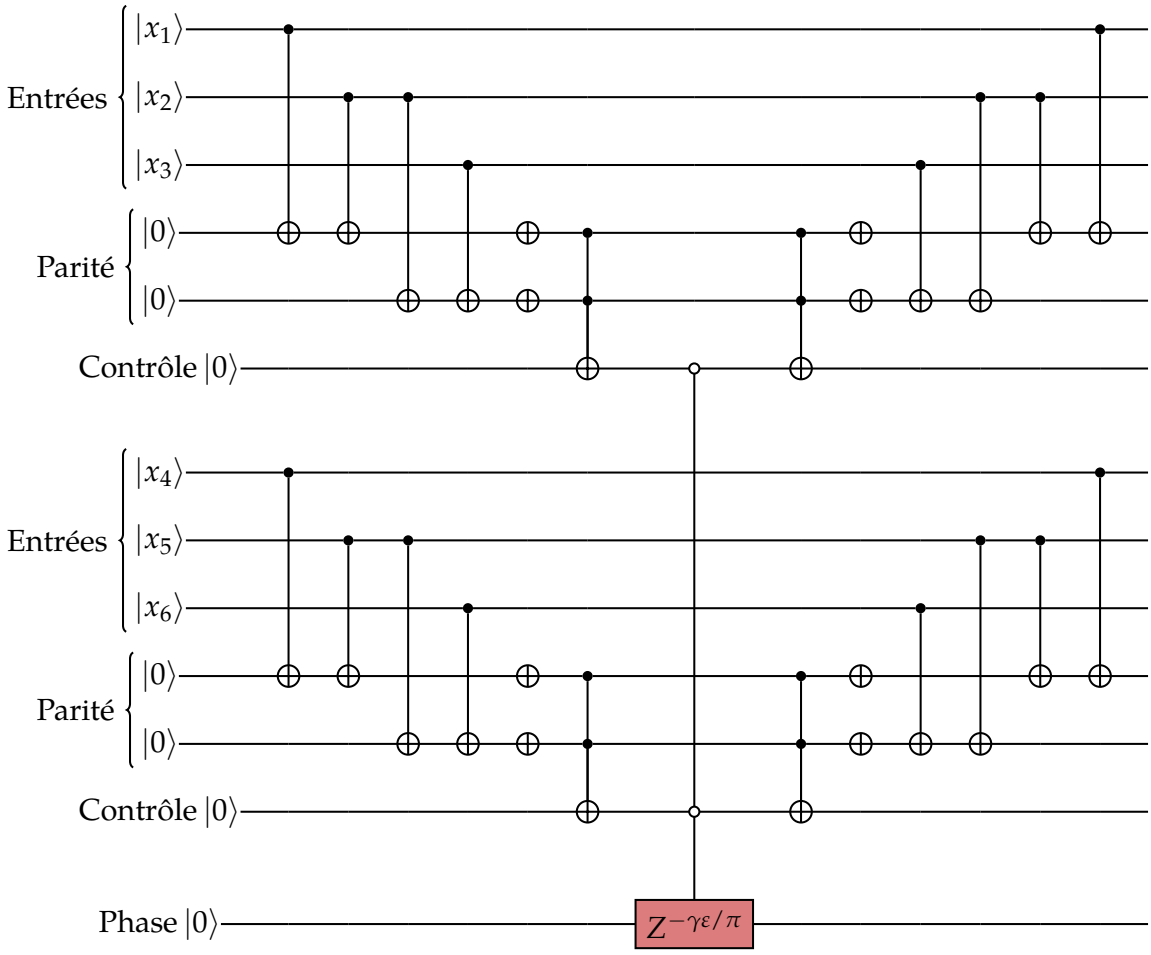

FIGURE 4.2 – Exemple d'un circuit d'effondrement des états excités. Pour une formule CNF $\varphi = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5 \vee x_6)$ décrivant un problème NAE3SAT, ce circuit applique l'opérateur $Z^{-\gamma\varepsilon/\pi}$ uniquement aux solutions du problème.

Chapitre 5

Résolution de problèmes #P-complet avec VQCount

Dans le chapitre précédent, un cadre théorique s'appuyant sur l'algorithme de JVV a été construit pour la résolution de problèmes de comptage à l'aide d'algorithmes variationnels quantiques. L'algorithme résultant, VQCount, obtient un compte approximatif à une erreur multiplicative près en échantillonnant un nombre polynomial de solutions à un circuit quantique optimisé GM-QAOA. Toutefois, plusieurs conditions potentiellement dispensables ou difficilement applicables ont été imposées. Ce chapitre change l'optique précédente en évaluant l'algorithme VQCount uniquement en tant que méthode heuristique, c'est-à-dire en caractérisant l'algorithme sans se soucier des garanties. La résolution de deux problèmes #P-complet, #NAE3SAT et #1in3SAT, est alors étudiée à l'aide de simulations de réseaux de tenseurs.

La section 5.1 définit les paramètres des simulations numériques effectuées et la section 5.2 explique la méthode utilisée pour caractériser l'algorithme VQCount. La performance de VQCount est examinée aux sections 5.3 et 5.4, où l'impact du biais d'échantillonnage des circuits QAOA et le comportement d'échelle sont respectivement présentés.

Le projet de ce mémoire a mené au développement d'un module python, nommé VQCount, disponible sur GitHub (`https://github.com/QuICoPhy-Lab/VQCount`). Ce module a été utilisé pour les simulations numériques de ce chapitre.

5.1 Paramètres de l'étude

Certaines précédentes considérations sont relâchées ou modifiées. D'abord, l'algorithme GM-QAOA n'est plus considéré dans sa forme limite de Grover. Plutôt qu'utiliser les paramètres fixes $\gamma_i = \beta_i = \pi$ pour $i = 1, \ldots, p$ dans le circuit GM-QAOA, les paramètres sont optimisés. De plus, les problèmes étudiés sont transformés à des hamiltoniens de problème contenant une tour d'états excités plutôt que de l'oracle à deux niveaux construit précédemment. Comme l'impact de la non-uniformité de la distribution d'état produite est étudié, l'algorithme VQCount est aussi employé avec l'algorithme quantique d'optimisation approximative. Cette section utilise donc QAOA pour référer à ce dernier. Finalement, la procédure d'auto-réduction de la section 4.2 est utilisée en renonçant à la réoptimisation des paramètres à chaque étape.

Pour les simulations numériques effectuées, les solutions sont échantillonnées sans remplacement. Ce choix s'écarte du protocole de l'algorithme de JVV, mais il accroît l'efficacité de la méthode lorsqu'un problème présente peu de solutions ou lorsque la distribution préparée par le circuit est fortement non-uniforme.

L'algorithme VQCount est appliqué aux problèmes #NAE3SAT positif et #1-in-3SAT positif présentés à la section 1.2. Le problème #NAE3SAT est difficile près du seuil de satisfaisabilité critique situé au rapport de clauses sur variables $\alpha_c \approx 2.1$. Pour étudier l'effet de la complexité des formules aléatoires sur la performance de VQCount, des instances sont générées à $\alpha = 1$ et $\alpha = 2$. Celles-ci sont générées à partir de graphes connectés bipartis, imposant une restriction supplémentaire. Similairement, les instances du problème positif #1-in-3SAT sont à leur seuil de difficulté le plus élevé près de $\alpha_c \approx 2/3$. Pour étudier ce problème dans le régime difficile, les instances sont générées à partir de graphes aléatoires cubiques, en plaçant une clause sur tous les sommets et une variable sur toutes les arêtes. Cette méthode permet d'échantillonner uniformément les instances difficiles, éliminant le biais dû à une sélection aléatoire [88]. Ce type de problème appartient à la catégorie des problèmes bloqués, étant vraisemblablement les plus difficiles des problèmes #P. Par la suite, les instances générées sont converties en un modèle d'Ising à l'aide de la transformation décrite à la section 3.2.1.

La simulation de circuits quantiques est un problème complexe en raison de la quantité de mémoire nécessaire pour représenter l'espace de Hilbert. La méthode la plus utilisée, la simulation à vecteur d'état, ne peut atteindre plus d'une vingtaine de qubits pour cette raison. Les réseaux de tenseurs sont un outil alternatif puissant, particulièrement pour l'étude de circuits quantiques peu profonds. L'annexe B introduit les réseaux de tenseurs et les méthodes particulières utilisées pour la simulation de l'algorithme VQCount. La librairie « quimb » [89] est utilisée pour modéliser l'optimisation et l'échantillonnage des circuits quantiques QAOA et GM-QAOA. Les paramètres de ces circuits sont optimisés avec la programmation séquentielle des moindres carrés, telle qu'implémentée par la librairie « SciPy » [90]. Pour évaluer la validité des résultats obtenus, « Ganak » [91], un compteur de modèles exact fondé sur une approche probabiliste, est utilisé pour déterminer le nombre exact de solutions. Le solveur SAT « Glucose4 » [92, 93], implémenté dans l'ensemble d'outils « PySAT » [94], est employé pour énumérer toutes les solutions possibles.

Pour le problème #NAE3SAT, 20 instances aléatoires sont générées par taille d'instance, allant de 6 à 20 variables, pour les deux densités de clause $\alpha = 1$ et $\alpha = 2$. Similairement, problème #1-in-3SAT, 20 instances aléatoires sont générées pour une taille de variable allant de 9 à 27 variables par saut de 3 variables, pour une densité de clause $\alpha = 2/3$.

5.2 Méthode

Afin de déterminer le nombre d'échantillons nécessaires pour atteindre un compte approximatif avec une tolérance ε et une confiance δ, le nombre d'échantillons utilisé par l'algorithme VQCount devrait être augmenté jusqu'à ce que le nombre de solutions trouvé soit à une erreur multiplicative ε du nombre de solutions exact avec une probabilité supérieure à δ. La simulation de l'algorithme étant coûteuse, une méthode plus simple et moins rigoureuse est utilisée ici. L'algorithme VQCount est exécuté en augmentant le nombre de solutions échantillonnées du circuit quantique optimisé QAOA jusqu'à ce que le nombre de solutions soit à l'intérieur des bornes d'erreur ε. La figure 5.1 montre la précision du comptage, c'est-à-dire le rapport du nombre de solutions approximatif sur le nombre de solutions exact, pour différentes

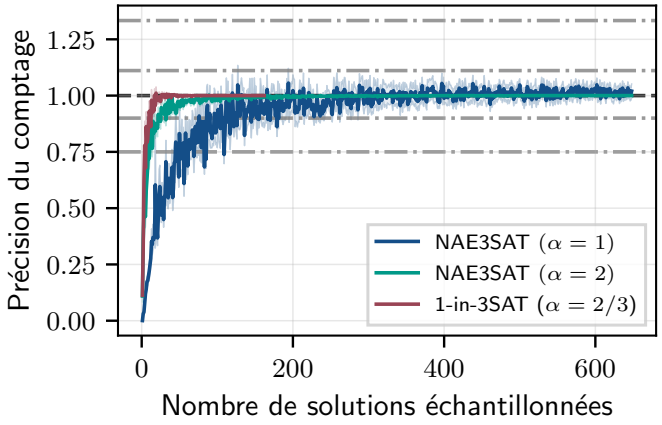

FIGURE 5.1 – Précision du comptage obtenue en augmentant le nombre de solutions échantillonnées à chaque étape de la procédure d'auto-réduction avec des circuits QAOA de profondeur $p = 3$ pour les instances NAE3SAT et 1-in-3SAT avec $n = 18$ variables. Les lignes solides représentent la moyenne et les régions ombragées montrent l'erreur standard de la moyenne. La ligne noire en pointillés représente la précision exacte du comptage, tandis que les lignes grises en tirets et pointillés correspondent à une erreur multiplicative de $\varepsilon = 1/3$ et $\varepsilon = 1/9$.

erreurs multiplicatives ε. Notons que, pour réduire les incertitudes, une procédure de lissage est utilisée. L'algorithme de JVV semble sous-estimer le nombre de solutions, comme vu avec la courbe de #NAE3SAT avec $\alpha = 1$. Ce comportement s'explique par le manque d'échantillons indiscernables dans la distribution de probabilité des solutions échantillonnées, entraînant une inexactitude des probabilités dans l'arbre d'auto-réductibilité.

5.3 Biais d'échantillonnage

Comme vu à la section 3.5, l'état préparé par un circuit GM-QAOA permet un échantillonnage uniforme de solutions tel que nécessaire pour l'algorithme de JVV, alors que la probabilité de certaines solutions est exponentiellement réduite avec l'approche QAOA. Sachant que les problèmes étudiés possèdent en moyenne un nombre exponentiel de solutions alors que seul un nombre polynomial d'échantillons est nécessaire, il est possible que la distribution préparée par QAOA soit suffisante pour obtenir un compte approximatif convenable. La figure 5.2 résume les

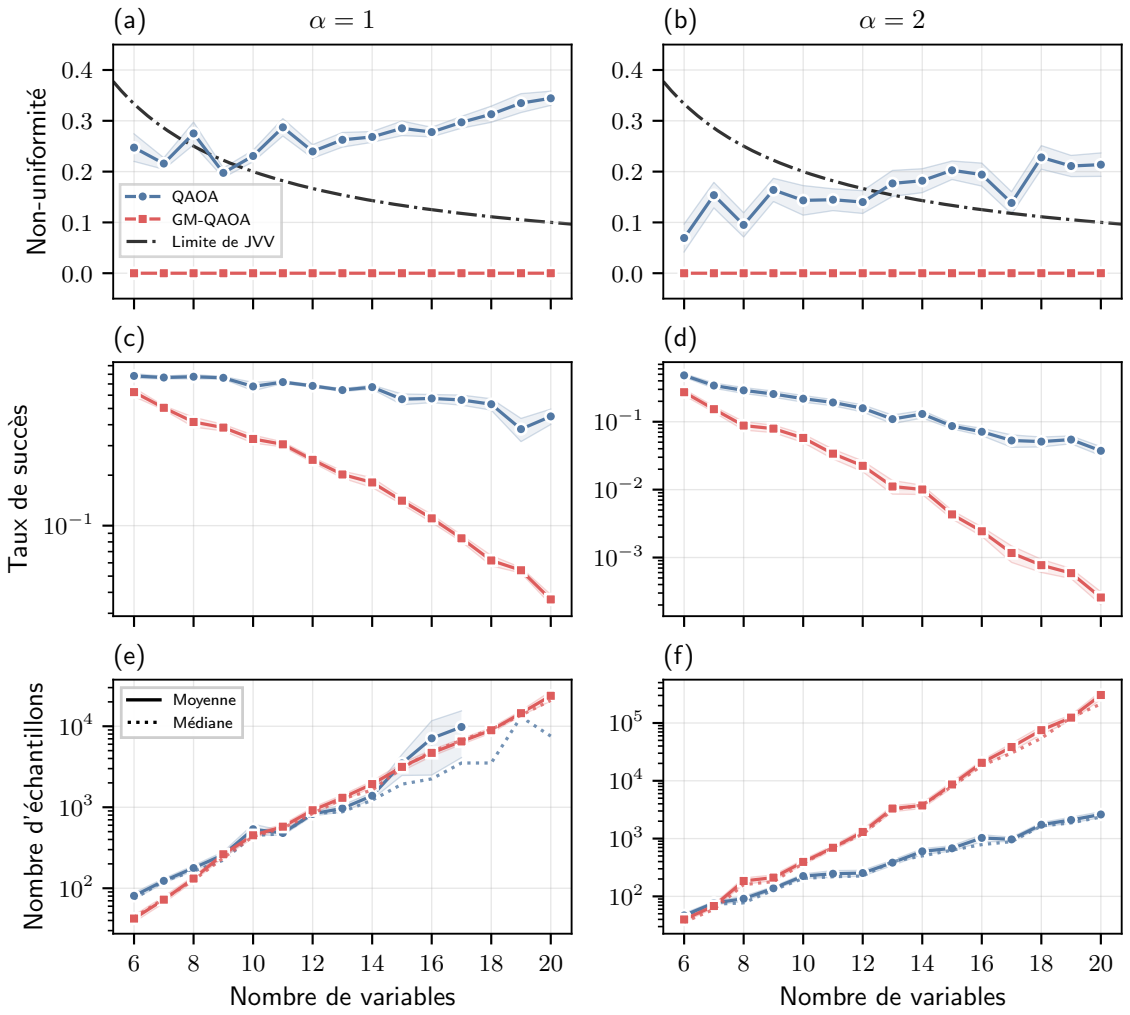

FIGURE 5.2 – Performance de VQCount avec des circuits QAOA de profondeur $p = 3$ (cercles bleus) et GM-QAOA (carrés rouges) pour les instances NAE3SAT à $\alpha = 1$ (panneaux de gauche) et $\alpha = 2$ (panneaux de droite). Les lignes solides et pointillées représentent respectivement la moyenne et la médiane. Les régions ombragées montrent l'erreur standard de la moyenne. (a,b) Non-uniformité maximale tout au long de la procédure d'auto-réduction. L'algorithme JVV nécessite que la non-uniformité soit au plus $O(n^{-1})$ pour n variables, comme illustré (lignes noires en tirets et pointillés) avec la fonction $f(n) = O(1/n)$. (c,d) Taux de succès minimal tout au long de la procédure d'auto-réduction. (e,f) Nombre d'échantillons nécessaires pour obtenir un comptage approximatif avec une tolérance d'erreur $\varepsilon = 1/3$. Un ajustement exponentiel extrapolé à partir des cinq derniers points de la médiane donne $O(1.45^n)$ (QAOA) et $O(1.44^n)$ (GM-QAOA) pour (e), et $O(1.34^n)$ (QAOA) et $O(1.88^n)$ (GM-QAOA) pour (f).

résultats obtenus pour le problème #NAE3SAT avec un circuit de profondeur $p = 3$. Les figures 5.2(a) et 5.2(b) confirment que GM-QAOA échantillonne les solutions parfaitement uniformément. Au contraire, la distribution des solutions préparées par QAOA est de plus en plus non-uniforme avec la taille de l'instance.

Les figures 5.2(c) et 5.2(d) montrent que, pour une profondeur de circuit fixe, le taux de succès de QAOA et GM-QAOA diminue grossièrement de façon exponentielle avec le nombre de variables, bien que la diminution est plus importante pour GM-QAOA. Les taux de succès sont plus faibles et se détériorent plus rapidement pour les instances avec $\alpha = 2$ qu'avec $\alpha = 1$, reflétant la différence de complexité de celles-ci selon leur proximité avec le seuil de satisfaisabilité.

Le nombre total d'échantillons requis pour que l'algorithme VQCount retourne un compte approximatif à une erreur multiplicative $\varepsilon = 1/3$ est présenté à la figure 5.2(e) et 5.2(f). Le nombre d'échantillons augmente exponentiellement avec la taille des instances en raison de la post-sélection des solutions à la fois pour QAOA et GM-QAOA comme générateurs de solutions. Bien que le taux de succès de GM-QAOA soit exponentiellement inférieur à QAOA, VQCount nécessite environ le même nombre d'échantillons avec QAOA ou GM-QAOA pour $\alpha = 1$. Au contraire, VQCount avec QAOA nécessite un nombre exponentiellement plus petit que celui de GM-QAOA, malgré la non-uniformité de la distribution des probabilités de QAOA. Cela suggère que QAOA est potentiellement un générateur plus efficace pour VQCount dans le cas où les solutions soient rares et dispersées. Le nombre d'échantillons post-sélectionnés requis pour que VQCount produise un estimé à un facteur multiplicatif ε près du compte exact, présenté à la figure 5.3, semble polynomial selon le nombre de variables n et l'inverse de la tolérance $1/\varepsilon$ comme attendu.

Sachant qu'un problème de plus petite taille est échantillonné à chaque étape de l'algorithme de JVV, il est raisonnable, du moins en théorie, de s'attendre à ce que le taux de succès de l'algorithme augmente. Cependant, dans cette étude, les paramètres du circuit quantique sont optimisés une seule fois et maintenus à des valeurs fixes au cours de la procédure d'auto-réduction, comme la réoptimisation est coûteuse à simuler classiquement. Est-ce que ce choix a un impact important sur la performance de l'algorithme VQCount ? Comme montré à la section 4.2, le

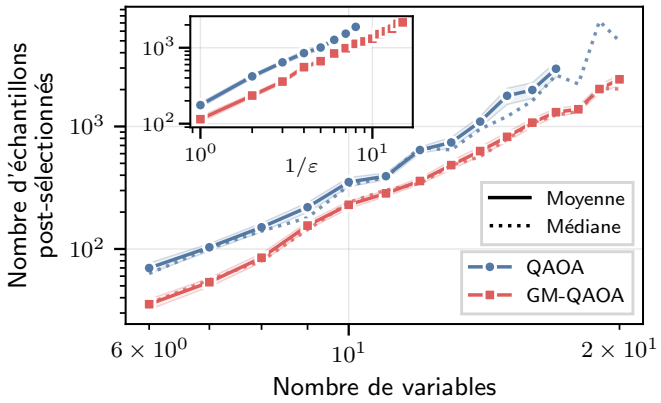

FIGURE 5.3 – Nombre d'échantillons post-sélectionnés nécessaires pour obtenir un comptage approximatif avec une tolérance d'erreur $\varepsilon = 1/3$ avec des circuits QAOA de profondeur $p = 3$ (cercles bleus) et GM-QAOA (carrés rouges) pour les instances NAE3SAT à $\alpha = 1$. Les lignes solides et pointillées représentent respectivement la moyenne et la médiane. Les régions ombragées montrent l'erreur standard de la moyenne. Un ajustement polynomial de la médiane donne $O(n^{5.12})$ (QAOA) et $O(n^{3.42})$ (GM-QAOA). Le panneau inséré montre le comportement de l'échelle en fonction de l'inverse de la tolérance d'erreur ε pour $n = 12$ variables. Un ajustement polynomial de la moyenne donne $O(\varepsilon^{-1.13})$ (QAOA) et $O(\varepsilon^{-1.07})$ (GM-QAOA).

taux de succès ne diminue pas avec le nombre d'étapes dans la limite de Grover. La figure 5.4 vérifie alors si cette hypothèse tient même en s'éloignant de cette limite. Les taux de succès demeurent effectivement non décroissants pour GM-QAOA. En revanche, la non-uniformité de QAOA augmente initialement avant de diminuer, comme vue dans la figure 5.4(a,b), avec une augmentation plus prononcée pour $\alpha = 1$. Ce comportement corrèle avec une diminution du taux de succès dans les premières étapes de l'auto-réduction. Le taux de succès de QAOA est toutefois toujours plus grand que celui de GM-QAOA.

Pour l'instant, la performance de VQCount a été étudiée pour des circuits de profondeur fixe. La figure 5.5 montre que, lorsque la profondeur du circuit augmente, la non-uniformité diminue et la probabilité d'obtenir une solution augmente, menant à un moindre nombre d'échantillons nécessaires pour atteindre un compte approximatif avec une erreur multiplicative donnée. QAOA semble s'améliorer plus rapidement selon la profondeur du circuit que GM-QAOA. D'autre part, la non-uniformité de QAOA augmente et atteint un plateau avec la profondeur du

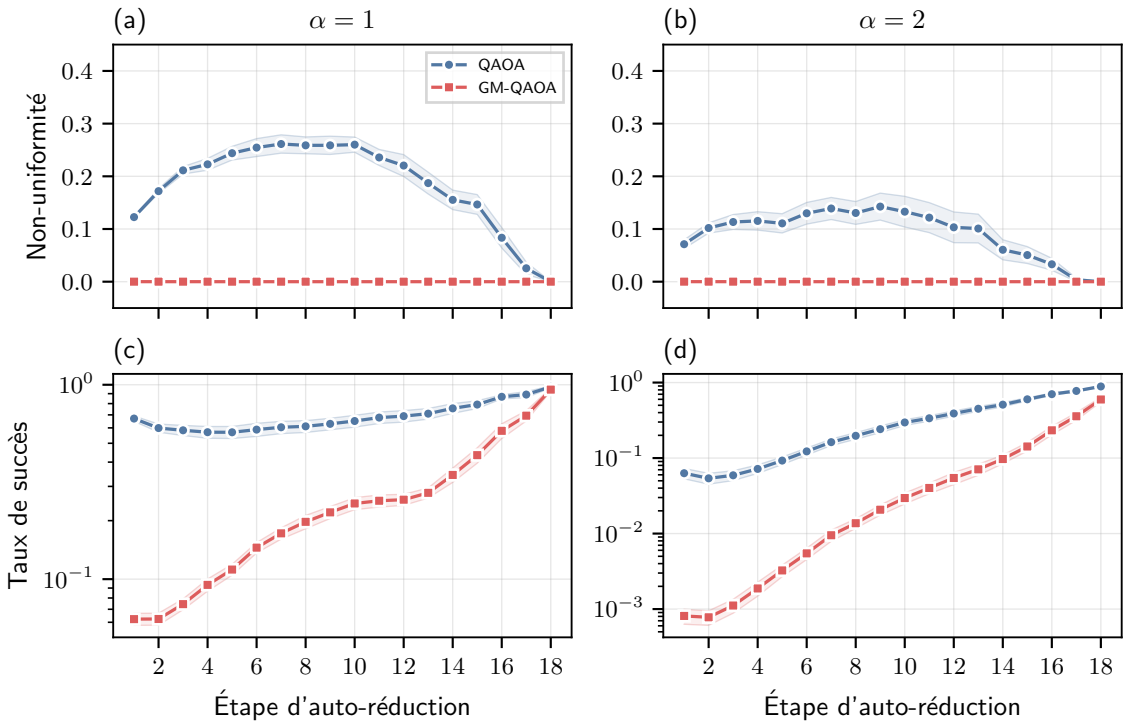

FIGURE 5.4 – Impact de la fixation des qubits tout au long de la procédure d'auto-réduction sur l'échantillonnage avec des circuits QAOA de profondeur $p = 3$ (cercles bleus) et GM-QAOA (carrés rouges) pour les instances NAE3SAT de $n = 18$ variables à $\alpha = 1$ (panneaux de gauche) et $\alpha = 2$ (panneaux de droite). Les lignes solides représentent la moyenne et les régions ombragées montrent l'erreur standard de la moyenne. (a,b) Non-uniformité en fonction du nombre d'étapes dans procédure d'auto-réduction. (c,d) Taux de succès en fonction du nombre d'étapes dans la procédure d'auto-réduction.

circuit.

Les résultats précédents suggèrent que le circuit QAOA est un meilleur générateur de solutions que le circuit GM-QAOA pour une faible profondeur. GM-QAOA dépend de plus d'une porte contrôlée à n qubits, difficilement à simuler classiquement et à réaliser avec les ordinateurs quantiques bruités à court terme. Ainsi, la section suivante se concentre sur les performances de VQCount à l'aide du générateur de solutions QAOA.

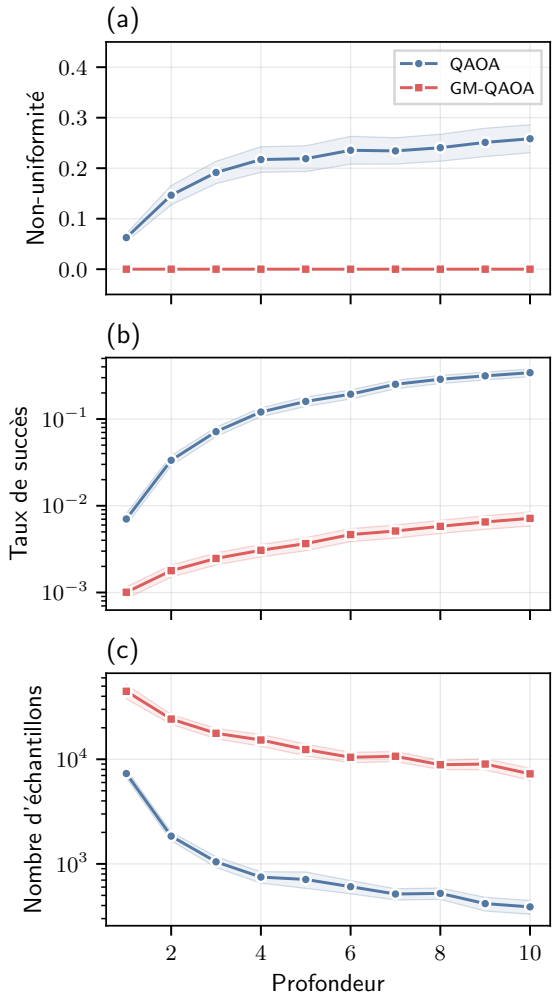

FIGURE 5.5 – Performance de VQCount en fonction de la profondeur avec des circuits QAOA (cercles bleus) et GM-QAOA (carrés rouges) pour les instances NAE3SAT de $n = 16$ variables à $\alpha = 2$. Les lignes solides représentent la moyenne et les régions ombragées montrent l'erreur standard de la moyenne. (a) Non-uniformité maximale tout au long de la procédure d'auto-réduction. (b) Taux de succès minimal tout au long de la procédure d'auto-réduction. (c) Nombre d'échantillons nécessaires pour obtenir un comptage approximatif avec une tolérance d'erreur $\varepsilon = 1/3$.

5.4 Comportement d'échelle

Ayant conclu que QAOA semble être un meilleur générateur de solutions pour une faible profondeur de circuit, la performance de VQCount avec QAOA est caractérisée de manière approfondie pour le problème #1-in-3SAT. La figure 5.6 résume

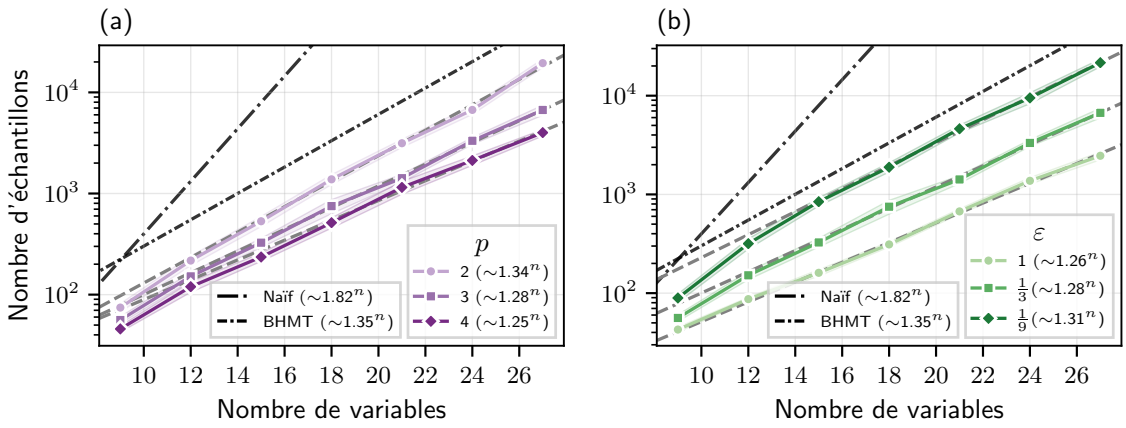

FIGURE 5.6 – Comportement de mise à l'échelle de VQCount avec des circuits QAOA pour les instances 1-in-3SAT à $\alpha = 2/3$. Le nombre d'échantillons nécessaires pour obtenir un comptage approximatif en utilisant l'échantillonnage par rejet naïf et l'algorithme de comptage quantique de Brassard, Høyer, Mosca et Tapp (BHMT) est indiqué respectivement par une ligne noire en tirets et pointillés et une ligne noire en tirets densément pointillés, suivant les fonctions $f(n) = O(1.82^n)$ et $f(n) = O(1.35^n)$ pour n variables. Les lignes solides représentent la moyenne, et les régions ombragées montrent l'erreur standard de la moyenne. Les lignes en pointillés indiquent l'ajustement exponentiel extrapolé à partir des quatre derniers points, comme indiqué dans la légende. (a) Nombre d'échantillons nécessaires pour que le comptage approximatif soit dans la tolérance d'erreur $\varepsilon = 1/3$ pour les profondeurs p. (b) Nombre d'échantillons nécessaires pour que le comptage approximatif soit dans la tolérance d'erreur ε pour une profondeur $p = 3$.

le comportement de mise à l'échelle de VQCount avec un circuit QAOA de faible profondeur. Naïvement, le nombre de solutions peut être trouvé en déterminant la probabilité qu'une chaîne de bit aléatoire échantillonnée de l'ensemble des chaînes de bits possible soit une solution. Comme vu à la figure 5.6(a), VQCount possède une loi d'échelle exponentiellement meilleure qu'avec l'échantillonnage par rejet naïf et sa performance s'améliore avec la taille du circuit. De plus, sa performance se compare à l'algorithme quantique de comptage approximatif de Brassard, Høyer, Mosca et Tapp [42], qui possède un comportement d'échelle $O(\sqrt{N/M})$ pour une taille d'espace de solutions M et une taille de l'espace de recherche total N. Aux profondeurs de circuits accessibles avec des ressources computationnelles raisonnables, l'efficacité asymptotique de VQCount est inférieure à celle des algorithmes classiques de pointe pour ce problème [35]. Cependant, les résultats suggèrent

que ce n'est possiblement pas le cas pour des profondeurs de circuit finies plus grandes. La précision de VQCount peut être améliorée en augmentant le nombre d'échantillons, comme montré à la figure 5.6(b). Finalement, la figure 5.7 montre le nombre d'échantillons post-sélectionnés nécessaire pour produire un compte à un facteur multiplicatif ε près du nombre de solutions exact en fonction du nombre de variables et de l'inverse de la tolérance. Ces résultats suggèrent un comportement d'échelle superpolynomial, mais sous-exponentiel du nombre d'échantillons post-sélectionnés nécessaires.

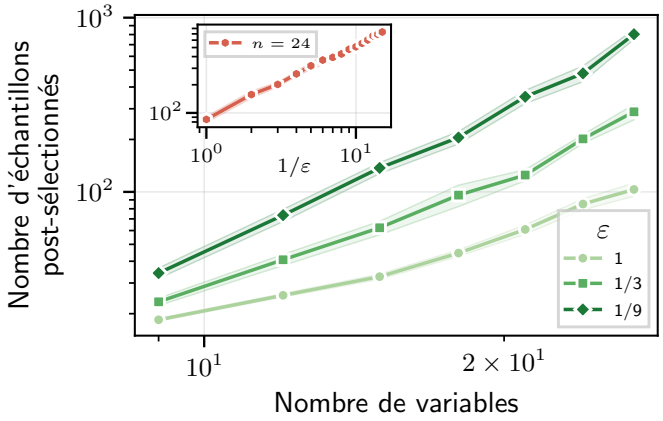

FIGURE 5.7 – Nombre d'échantillons post-sélectionnés nécessaires pour obtenir un comptage approximatif avec une tolérance d'erreur ε avec des circuits QAOA de profondeur $p = 3$ pour les instances 1-in-3SAT à $\alpha = 2/3$. Le panneau inséré montre le comportement de mise à l'échelle en fonction de l'inverse de la tolérance d'erreur ε pour $n = 24$ variables, pour lequel un ajustement polynomial donne $O(\varepsilon^{-0.79})$. Les lignes solides représentent la moyenne, et les régions ombragées montrent l'erreur standard de la moyenne.

L'algorithme VQCount permet d'estimer un nombre exponentiel de solutions en utilisant seulement un nombre polynomial d'échantillons. Toutefois, dans les simulations numériques, certaines instances de problème peuvent comporter un petit nombre de solutions. Il est donc essentiel de vérifier que VQCount requiert, en pratique, moins d'échantillons que le nombre total de solutions, afin d'éviter une simple énumération exhaustive. La figure 5.8 illustre l'efficacité d'échantillonnage de VQCount, définie comme le rapport entre le nombre exact de solutions et le nombre de solutions distinctes utilisées par l'algorithme, pour les deux problèmes étudiés

dans ce travail. L'efficacité d'échantillonnage est positivement corrélée à la densité des solutions. Néanmoins, VQCount utilise systématiquement moins d'échantillons que le nombre total de solutions.

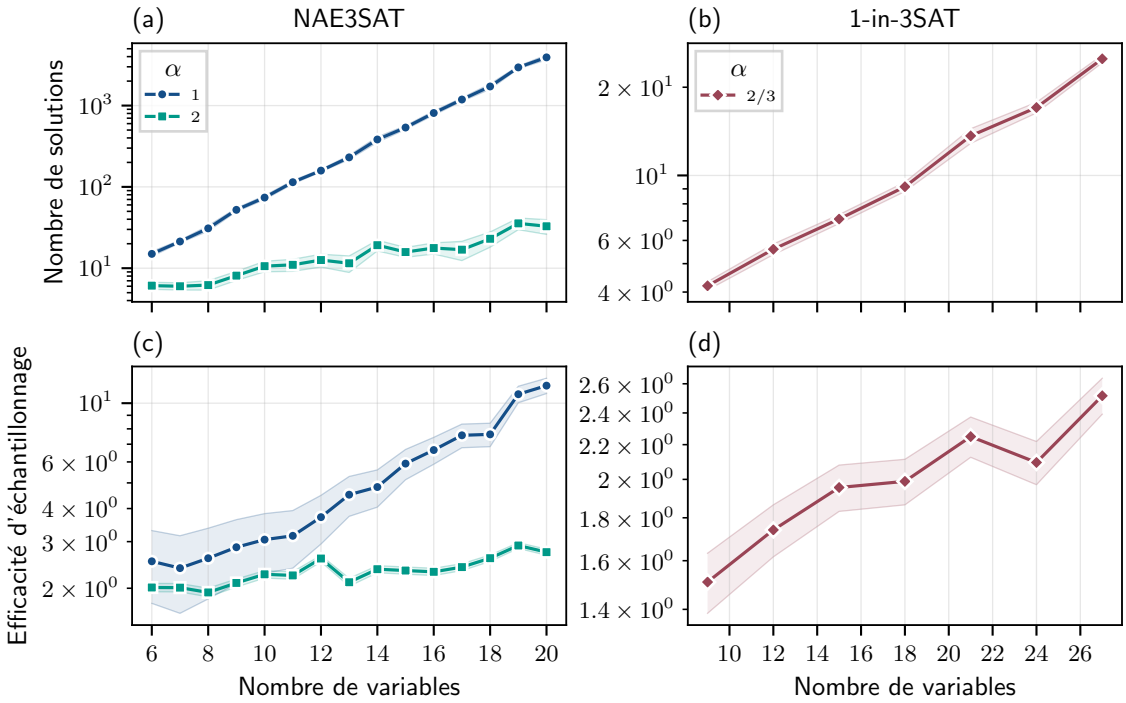

FIGURE 5.8 – Nombre de solutions (a) et efficacité d'échantillonnage (b) pour atteindre une tolérance d'erreur $\varepsilon = 1$ avec des circuits QAOA de profondeur $p = 3$ pour les instances NAE3SAT (panneaux de gauche) et 1-in-3SAT (panneaux de droite). Les lignes solides représentent la moyenne, et les régions ombragées montrent l'erreur standard de la moyenne.

Conclusion

Ce travail introduit VQCount, un algorithme variationnel quantique basé sur l'algorithme de Jerrum, Valiant et Vazirani pour le comptage approximatif de modèles à partir de l'échantillonnage aléatoire de l'état préparé par l'algorithme quantique à opérateurs alternants. La validité de VQCount est établie dans une limite où la forme de l'algorithme de Grover est retrouvée. Comparativement aux travaux précédents, VQCount nécessite un nombre d'échantillons exponentiellement plus petit pour obtenir une estimation à un facteur multiplicatif près du compte exact lorsque le nombre de solutions est exponentiel. La performance de VQCount est étudiée numériquement comme méthode heuristique, en utilisant la contraction de réseaux de tenseurs pour la simulation des circuits quantiques composant l'algorithme. Pour les problèmes de dénombrement #NAE3SAT et #1-in-3SAT, un compromis est établi entre le taux de succès et l'uniformité de l'échantillonnage de solutions. De plus, un gain exponentiel du nombre d'échantillons nécessaire est observé en comparaison à l'échantillonnage par rejet naïf.

L'algorithme VQCount introduit la fondation d'un algorithme variationnel quantique pour le comptage. La seule technique de post-traitement utilisée est le rejet des échantillons n'appartenant pas à l'ensemble de solutions. Toutefois, des techniques plus élaborées, comme la mise à jour de clusters sans rejet à température nulle [95], pourraient être employées. Les algorithmes de comptage approximatif basés sur l'échantillonnage aléatoire ont aussi grandement évolué depuis le travail de Jerrum, Valiant et Vazirani. Par exemple, l'algorithme introduit par ces derniers peut être modifié pour obtenir des garanties sur le compte approximatif obtenu même pour des distributions de solutions non-uniforme [96]. Ces nouvelles techniques pourraient être employées pour rivaliser avec les algorithmes classiques modernes.

Annexe A

Expression fermée de l'état du circuit GM-QAOA

Théorème A.1 – Expression fermée de l'état du circuit GM-QAOA

Pour un espace de solutions non contraint, l'état préparé par un circuit GM-QAOA $|\psi(\vec{\gamma},\vec{\beta})\rangle$ de profondeur p pour n qubits s'exprime sous la forme de l'expression récursive fermée suivante :

$$|\psi\rangle = \frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_p\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}\bar{F}_p^{(k)}\,|e^{(k)}\rangle\right), \tag{A.1}$$

où $N = 2^n$, $F_0 = \bar{F}_0^{(k)} = 1$ et

$$F_p = F_{p-1} - \frac{1}{N}(1-e^{-i\beta_p})\left(|G|F_{p-1} + \sum_{k'}|E^{(k')}|\bar{F}_{p-1}^{(k')}e^{-i\gamma_p\varepsilon_{k'}}\right), \tag{A.2}$$

$$\bar{F}_p^{(k)} = e^{-i\gamma_p\varepsilon_k}\bar{F}_{p-1}^{(k)} - \frac{1}{N}(1-e^{-i\beta_p})\left(|G|F_{p-1} + \sum_{k'}|E^{(k')}|\bar{F}_{p-1}^{(k')}e^{-i\gamma_p\varepsilon_{k'}}\right), \tag{A.3}$$

pour les états fondamentaux $|g\rangle \in G$ et les états excités $|e^{(k)}\rangle \in E^{(k)}$ de niveau k.

Démonstration. Soit φ l'instance d'un problème de #P de taille n que nous voulons résoudre avec GM-QAOA. Cette instance peut être encodée dans un hamiltonien de problème H_P avec des valeurs propres $g = 0$ pour les solutions et $e^{(k)} = \varepsilon_k \in \mathbb{R}_{>0}$

pour les non-solutions. L'opérateur U_P correspondant avec H_P s'écrit alors comme

$$U_P = \sum_{j \in G} |j\rangle \langle j| + \sum_k e^{-i\gamma\varepsilon_k} \sum_{j \in E^{(k)}} |j\rangle \langle j| \,, \tag{A.4}$$

où G et $E^{(k)}$ sont respectivement les variétés des états fondamentaux et états excités, et donc l'ensemble des solutions et des non-solutions de φ. Pour un espace de solutions non contraint, l'opérateur U_D^{GM} est donné par

$$U_D^{GM} = \mathbb{1} - (1 - e^{-i\beta})(|+\rangle\langle+|^{\otimes n}) \,. \tag{A.5}$$

Les états fondamentaux et excités s'écrivent comme

$$|g\rangle = \frac{1}{\sqrt{|G|}} \sum_{j \in G} |j\rangle \,, \tag{A.6}$$

$$|e^{(k)}\rangle = \frac{1}{\sqrt{|E^{(k)}|}} \sum_{j \in E^{(k)}} |j\rangle \,. \tag{A.7}$$

Montrons, par induction, que l'état final suit la formule récursive A.1 en évoluant l'état initial $|\psi_0\rangle = |+\rangle^{\otimes n}$ avec un circuit GM-QAOA de p couches.

Cas 1 (1 couche) : Calculons $|\psi_1\rangle = U_P(\gamma_1)\,|\psi_0\rangle$.

$$\begin{aligned} |\psi_1\rangle &= \left(\sum_{j \in G} |j\rangle \langle j| + \sum_k e^{-i\gamma_1\varepsilon_k} \sum_{j \in E^{(k)}} |j\rangle \langle j| \right) \left(\frac{1}{\sqrt{N}} \sum_{j'=0}^{N-1} |j'\rangle \right) \\ &= \frac{1}{\sqrt{N}} \left(\sum_{j \in G} |j\rangle + \sum_k e^{-i\gamma_1\varepsilon_k} \sum_{j \in E^{(k)}} |j\rangle \right) \\ &= \frac{1}{\sqrt{N}} \left(\sqrt{|G|}\,|g\rangle + \sum_k \sqrt{|E^{(k)}|} e^{-i\gamma_1\varepsilon_k} |e^{(k)}\rangle \right) , \end{aligned} \tag{A.8}$$

où la relation $\sum_{j \in G} |j\rangle \langle j| \sum_{j'=0}^{N} |j'\rangle = \sum_{j \in G} |j\rangle$ a été utilisée.

Calculons désormais $|\psi_2\rangle = U_D(\beta_1)\,|\psi_1\rangle$.

$$\begin{aligned}
|\psi_2\rangle &= \left(\mathbb{1} - \frac{1}{N}(1-e^{-i\beta_1})\sum_{i',j'=0}^{N}|i'\rangle\langle j'|\right)\left(\frac{1}{\sqrt{N}}\left(\sqrt{|G|}\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}e^{-i\gamma_1\varepsilon_k}\,|e^{(k)}\rangle\right)\right)\\
&= \frac{1}{\sqrt{N}}\left(\sqrt{|G|}\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}e^{-i\gamma_1\varepsilon_k}\,|e^{(k)}\rangle\right.\\
&\left.-\frac{1}{N}(1-e^{-i\beta_1})\left(|G| + \sum_k |E^{(k)}|e^{-i\gamma_1\varepsilon_k}\right)\sum_{i'=0}^{N}|i'\rangle\right)\\
&= \frac{1}{\sqrt{N}}\left(\sqrt{|G|}\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}e^{-i\gamma_1\varepsilon_k}\,|e^{(k)}\rangle\right.\\
&\left.-\frac{1}{N}(1-e^{-i\beta_1})\left(|G| + \sum_k |E^{(k)}|e^{-i\gamma_1\varepsilon_k}\right)\left(\sqrt{|G|}\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}\,|e^{(k)}\rangle\right)\right)\\
&= \frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_1\,|g\rangle + \sum_k \sqrt{|E^{(k)}|}\bar{F}^{(k)}\,|e^{(k)}\rangle\right),
\end{aligned} \tag{A.9}$$

où

$$F_1 = 1 - \frac{1}{N}(1-e^{-i\beta_1})\left(|G| + \sum_{k'}|E^{(k')}|e^{-i\gamma_1\varepsilon_{k'}}\right), \tag{A.10}$$

$$\bar{F}_1^{(k)} = e^{-i\gamma_1\varepsilon_k} - \frac{1}{N}(1-e^{-i\beta_1})\left(|G| + \sum_{k'}|E^{(k')}|e^{-i\gamma_1\varepsilon_{k'}}\right). \tag{A.11}$$

Le cas initial a donc été prouvé.

Cas 2 (p couche) : Supposons que $|\psi\rangle = U_D(\beta_p)U_P(\gamma_p)\ldots U_D(\beta_1)U_P(\gamma_1)\,|\psi_0\rangle$ donne lieu à la relation suivante

$$|\psi\rangle = \frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_p\,|g_i\rangle + \sum_k \sqrt{|E^{(k)}|}\bar{F}_p^{(k)}\,|e^{(k)}\rangle\right), \tag{A.12}$$

où

$$\begin{aligned}
F_p &= F_{p-1} - \frac{1}{N}(1-e^{-i\beta_p})\left(|G|F_{p-1} + \sum_{k'}|E^{(k')}|\bar{F}_{p-1}^{(k')}e^{-i\gamma_p\varepsilon_{k'}}\right),\\
\bar{F}_p^{(k)} &= e^{-i\gamma_p\varepsilon_k}\bar{F}_{p-1}^{(k)} - \frac{1}{N}(1-e^{-i\beta_p})\left(|G|F_{p-1} + \sum_{k'}|E^{(k')}|\bar{F}_{p-1}^{(k')}e^{-i\gamma_p\varepsilon_{k'}}\right).
\end{aligned} \tag{A.13}$$

Cas 3 ($p+1$ couche) : En utilisant la supposition précédente, appliquons U_P en calculant $|\psi_1\rangle = U_P |\psi_0\rangle$.

$$\begin{aligned}
|\psi_1\rangle &= \left(\sum_{j\in G} |j\rangle\langle j| + \sum_k e^{-i\gamma_{p+1}\varepsilon_k} \sum_{j\in E^{(k)}} |j\rangle\langle j|\right)\left(\frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_p|g\rangle + \sum_k \sqrt{|E^{(k)}|}\bar{F}_p^{(k)}|e^{(k)}\rangle\right)\right)\\
&= \frac{1}{\sqrt{N}}\left(\sum_{j\in G_i} |j\rangle\langle j| + \sum_{i=0}^{Q-1}\sum_k e^{-i\gamma_{p+1}\varepsilon_k}\sum_{j\in E_i^{(k)}} |j\rangle\langle j|\right)\left(F_p\sum_{j\in G}|j\rangle + \sum_k \bar{F}_p^{(k)}\sum_{j\in E^{(k)}}|j\rangle\right)\\
&= \frac{1}{\sqrt{N}}\left(F_p\sum_{j\in G}|j\rangle + \sum_k \bar{F}_p^{(k)} e^{-i\gamma_{p+1}\varepsilon_k}\sum_{j\in E^{(k)}}|j\rangle\right)\\
&= \frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_p|g\rangle + \sum_k\sqrt{|E^{(k)}|}\bar{F}_p^{(k)}e^{-i\gamma_{p+1}\varepsilon_k}|e^{(k)}\rangle\right).
\end{aligned} \tag{A.14}$$

Appliquons ensuite U_D en calculant $|\psi_2\rangle = U_D|\psi_1\rangle$.

$$\begin{aligned}
|\psi_2\rangle &= \left(\mathbb{1} - \frac{1}{N}(1-e^{-i\beta_{p+1}})\sum_{i',j'=0}^{N}|i'\rangle\langle j'|\right)\\
&\quad\left(\frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_p|g\rangle + \sum_k\sqrt{|E^{(k)}|}\bar{F}_p^{(k)}e^{-i\gamma_{p+1}\varepsilon_k}|e^{(k)}\rangle\right)\right)\\
&= \frac{1}{\sqrt{N}}\left(\left(\sqrt{|G|}F_p|g\rangle + \sum_k\sqrt{|E^{(k)}|}\bar{F}_p^{(k)}e^{-i\gamma_{p+1}\varepsilon_k}|e^{(k)}\rangle\right)\right.\\
&\quad\left.-\frac{1}{N}(1-e^{-i\beta_{p+1}})\left(|G|F_p + \sum_k|E^{(k)}|\bar{F}_p^{(k)}e^{-i\gamma_{p+1}\varepsilon_k}\right)\sum_{i'=0}^{N}|i'\rangle\right)\\
&= \frac{1}{\sqrt{N}}\left(\sqrt{|G|}F_{p+1}|g_i\rangle + \sum_k\sqrt{|E^{(k)}|}\bar{F}_{p+1}^{(k)}e^{-i\gamma_{p+1}\varepsilon_k}|e_i^{(k)}\rangle\right),
\end{aligned} \tag{A.15}$$

où

$$\begin{aligned}
F_{p+1} &= F_p - \frac{1}{N}(1-e^{-i\beta_{p+1}})\left(|G|F_p + \sum_{k'}|E^{(k')}|\bar{F}_p^{(k')}e^{-i\gamma_{p+1}\varepsilon_{k'}}\right),\\
\bar{F}_{p+1}^{(k)} &= e^{-i\gamma_{p+1}\varepsilon_k}\bar{F}_p^{(k)} - \frac{1}{N}(1-e^{-i\beta_{p+1}})\left(|G|F_p + \sum_{k'}|E^{(k')}|\bar{F}_p^{(k')}e^{-i\gamma_{p+1}\varepsilon_{k'}}\right).
\end{aligned} \tag{A.16}$$

La formule récursive a donc été prouvée par induction. □

Annexe B

Simulation de circuits quantiques avec les réseaux de tenseurs

La simulation classique d'états quantiques est tout sauf évidente. Ces états, appartenant à l'espace de Hilbert, sont décrit par des vecteurs d'état de taille exponentielle empêchant ainsi leur caractérisation même pour des systèmes de taille modeste. Cette difficulté, présente dans de nombreux domaines, tel l'apprentissage automatique, est connue sous le nom de la *malédiction de la dimensionnalité*. Cependant, la représentation de certains états physiques intéressants contient parfois de l'information superflue ou une structure inhérente. Par exemple, un état quantique général de n qubits nécessite théoriquement un vecteur d'état à 2^n bits, mais un état quantique non intriqué nécessite seulement $2n$ bits en raison de l'absence de corrélation. Cette idée a alors mené au développement des méthodes de réseaux de tenseurs pour l'étude de système quantique à plusieurs corps en matière condensée afin d'obtenir une représentation des états quantiques plus efficaces.

Les réseaux de tenseurs font partie des méthodes les plus puissantes pour la simulation de circuits quantiques. Comme ceux-ci sont utilisés pour les simulations numériques présentées au chapitre 5, une brève introduction à la matière est nécessaire. Cette annexe décrit seulement en surface les méthodes de réseaux de tenseurs. Pour comprendre les différentes méthodes plus en profondeur, de nombreux excellents tutoriels ont été écrits [97-99]. La section B.1 introduit les réseaux de tenseurs ainsi que les principales opérations possibles. Une représentation alternative de ceux-ci, les états en produit de matrices, est décrite à la section B.2 alors que la correspondance entre la simulation de circuits quantiques et la contraction de réseaux de tenseurs est expliquée à la section B.3.

B.1 Réseaux de tenseurs

Un *tenseur* est un tableau multilinéaire de rang, c'est-à-dire un nombre de dimensions, arbitraire encodant une certaine quantité d'information. Un tenseur s'interprète aussi comme une fonction définie sur un domaine discret. Par exemple, un scalaire, un vecteur et une matrice correspondent respectivement à un tenseur de rang 0, 1 et 2. Les tenseurs généralisent en outre ces derniers objets en admettant des tenseurs de rang m.

Définition B.1 – Tenseur

Un tenseur $T[x_1, x_2, \ldots, x_m]$ de rang m avec dimensions $d_1 \times d_2 \times \cdots \times d_m$ est un élément de l'ensemble $\mathbb{C}^{d_1 \times d_2 \times \cdots \times d_m}$.

Pour simplifier la notation, un tenseur s'écrit plus succinctement par $T[\varepsilon]$ où $\varepsilon \subseteq \{ x_1, x_2, \ldots, x_m \}$. L'attrait principal des tenseurs est leur représentation géométrique simple, qui permet de visualiser aisément leurs caractéristiques principales ainsi que de faciliter leur manipulation. Tout tenseur T se représente par une forme géométrique, où chaque indice x_i est représenté par une patte sortant de la forme telle qu'illustrée à la figure B.1. Un réseau de tenseurs est une collection de tenseurs $\{ T_1[\varepsilon_1], T_2[\varepsilon_2], \ldots, T_n[\varepsilon_n] \}$ partageant certains indices x_i. La notation en réseaux de tenseurs s'apparente beaucoup à la notation d'Einstein ; les sommations sont donc souvent implicites.

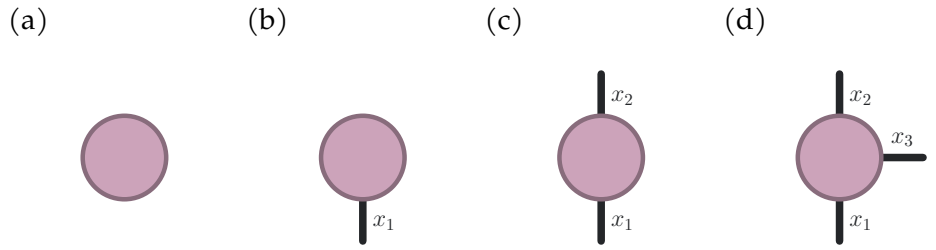

FIGURE B.1 – Représentation géométrique de tenseurs de différents rangs. Des tenseurs de rang 0 (scalaire), 1 (vecteur), 2 (matrice) et 3 (tenseur de rang 3), sont illustrés respectivement aux panneaux (a), (b), (c) et (d).

Plusieurs opérations peuvent être appliquées sur les tenseurs constituants d'un réseau de tenseurs. Le *produit tensoriel* entre deux tenseurs A et B sur les ensembles d'indices respectifs ε_1 et ε_2 correspond au produit élément par élément des valeurs des tenseurs, généralisant ainsi le produit extérieur entre deux vecteurs

$$(A \otimes B)[\varepsilon_1, \varepsilon_2] \coloneqq A[\varepsilon_1] \cdot B[\varepsilon_2]\,. \tag{B.1}$$

Le produit tensoriel se visualise simplement par la juxtaposition des deux tenseurs. La *trace* (partielle) d'un tenseur A est la sommation jointe sur deux indices x_i et x_j de A ayant la même dimension $d_{x_i} = d_{x_j}$

$$\mathrm{Tr}_{x_i,x_j}(A[x_i, x_j, \varepsilon \setminus \{\, x_i, x_j \,\}]) \coloneqq \sum_{k}^{d_{x_i}} A[k, k, \varepsilon \setminus \{\, x_i, x_j \,\}]\,. \tag{B.2}$$

Géométriquement, la trace implique de joindre les pattes du tenseur. La *contraction* de deux tenseurs est l'opération la plus commune, consistant en un produit tensoriel entre deux tenseurs A et B suivi d'une trace entre les indices communs $\partial\varepsilon$ de ces derniers

$$C[\varepsilon \setminus \{\, \partial\varepsilon \,\}] \coloneqq \sum_{\partial\varepsilon} (A \otimes B)[\partial\varepsilon, \varepsilon \setminus \{\, \partial\varepsilon \,\}] \tag{B.3}$$

Deux tenseurs contractés se combinent alors en une seule forme géométrique avec des pattes correspondant aux indices non contractés. Comme les tenseurs ne sont en réalité que des tableaux multidimensionnels, il est possible de réorganiser l'information encodée sous une différente forme. Les opérations de *regroupement* et de *séparation* modifient l'organisation de cette information. Par exemple, une matrice représentée par une matrice de taille $d_1 \times d_2$ peut aussi être encodée dans un vecteur de taille $d_1 \cdot d_2$. Cette manipulation des données se représente graphiquement par un regroupement ou une séparation des pattes des tenseurs.

Finalement, l'opération inverse à la contraction est la *décomposition*, où un tenseur est décomposé en deux différents tenseurs. Un exemple particulièrement intéressant est la *décomposition en valeurs singulières*, une méthode typiquement appliquée aux matrices, mais pouvant se généraliser aux tenseurs en les regroupant sous forme de matrices. Cette décomposition permet de plus de réduire la dimension des indices et donc de retirer l'information superflue en retirant les valeurs négligeables de la matrice singulière.

Les méthodes de réseaux de tenseurs suivent couramment la démarche suivante. Le problème à résoudre est d'abord modélisé sous la forme d'un réseau de tenseurs en encodant les contraintes sous la forme de tenseurs, de façon à ce que la contraction de ce réseau donne la solution au problème. En utilisant les opérations précédentes, incluant potentiellement les opérations de décomposition pour la simplification, le réseau est contracté pour obtenir la solution souhaitée.

Notons que l'ordre de contraction, c'est-à-dire l'ordre dans lequel chacun des tenseurs est contracté avec les tenseurs environnants, à un fort impact sur la quantité de ressources nécessaires [100].

B.2 État en produit de matrices et opérateur en produit de matrices

Un *état en produit de matrices* (« Matrix Product State ») (MPS) est un type particulier de réseau de tenseurs permettant la représentation efficace d'un état quantique. Considérons un état quantique $|\psi\rangle$ à n qubits. En employant la notation en réseau de tenseurs, cet état se représente par un tenseur de rang n, avec n indices de dimensions $d = 2$, encodant les amplitudes des qubits de $|\psi\rangle$. Cette représentation nécessite un grand nombre d'éléments, plus précisément 2^n, pour encoder l'entièreté des amplitudes des qubits. Alternativement, ce tenseur peut être décomposé en n tenseurs à l'aide de la décomposition en valeurs singulières. Ce nouveau réseau de tenseurs, illustré à la figure B.2a, représente alors un MPS. En coupant les plus petites valeurs de la matrice singulière lors de la décomposition, correspondant à l'information redondante du système, une représentation de taille réduite peut être obtenue, permettant ainsi l'étude de l'état souhaité. Les MPS sont particulièrement utiles pour l'étude de l'intrication entre deux sous-parties d'un système comme celle-ci est capturée dans les indices latéraux du MPS. Notons que la taille de la représentation sous forme de MPS est grandement dépendante de l'intrication du système. Un système fortement intriqué ne possède pas de représentation efficace à l'aide de ce type d'objet.

Les opérateurs en produit de matrices permettent quant à eux d'appliquer un opérateur sur un MPS. Ces opérateurs empruntent la forme d'un MPS en rajoutant une patte par tenseur tel qu'illustré à la figure B.2b. La méthode MPS/MPO peut par exemple représenter l'évolution d'un système quantique selon un certain hamiltonien (représenté sous forme de MPO). Ces opérateurs permettent de plus de représenter les matrices de densités. Encore une fois, l'utilité de ceux-ci dépend grandement de l'intrication qu'ils ajoutent au MPS.

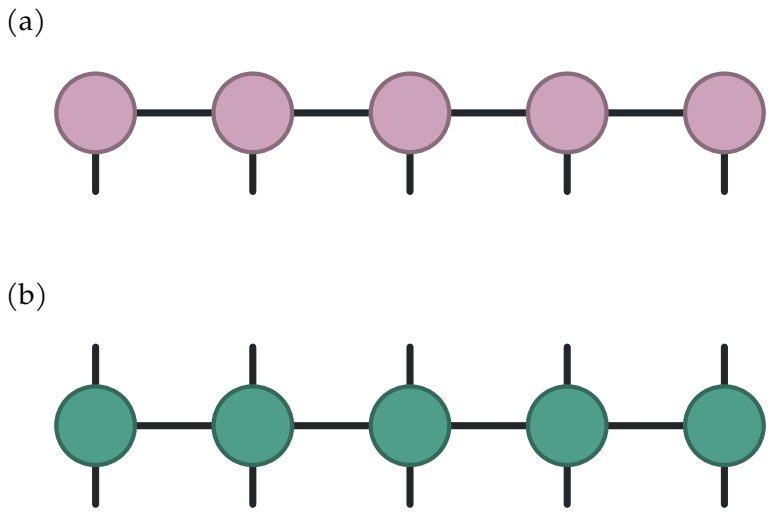

FIGURE B.2 – Schéma d'un état (a) et d'un opérateur (b) en produit de matrices.

B.3 Simulation de circuits quantiques

Les réseaux de tenseurs font partie des méthodes les plus performantes pour la simulation de circuits quantiques. Il y a en effet une correspondance parfaite entre les réseaux de tenseurs et les circuits quantiques. L'état d'un qubit s'encode facilement par un tenseur de rang $d = 1$. Par exemple, pour un qubit dans une superposition égale, l'état est représenté par le tenseur

$$T^{|\psi\rangle} = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} . \tag{B.4}$$

De plus, les opérateurs d'un circuit quantique se représentent aussi simplement par des tenseurs. Par exemple, une porte de Pauli X correspond à un tenseur

$$T^{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{B.5}$$

de rang $d = 2$. La contraction d'un réseau composé des différents opérateurs d'un circuit donné est alors équivalente à l'état obtenu par l'application d'opérateurs quantiques. La méthode MPS/MPO se prête bien à la simulation de circuit quantique, particulièrement lorsque l'intrication de celui-ci demeure faible. Les simulations effectuées au cours du travail de ce mémoire emploient alors autant la contraction de réseaux de tenseurs sans forme précise que la contraction de MPS/MPO, selon l'efficacité de la méthode pour un problème étudié.

Afin d'obtenir des échantillons à partir d'un réseau de tenseurs, deux méthodes sont possibles. D'abord, une représentation dense de la fonction d'onde peut être obtenue en contractant complètement le réseau. Les configurations sont alors simplement obtenues en pigeant selon la distribution de probabilité trouvée. Cette stratégie exige de sauvegarder un nombre exponentiel de valeurs, ce qui est infaisable pour des instances de grandes tailles. Une méthode alternative est plutôt la méthode proposée par Ferris et Vidal [101]. Pour mesurer la valeur moyenne d'opérateurs locaux, les cônes causaux passés de ceux-ci sont utilisés pour éviter de contracter l'entièreté du réseau de tenseurs. Une séquence de matrices de densité conditionnelles est alors calculée en contractant un à un les opérateurs du cône causal pour permettre l'échantillonnage de la distribution de probabilités des configurations. Cette approche garantit un échantillonnage parfait, c'est-à-dire qu'il est possible d'obtenir des échantillons totalement non corrélés directement de la distribution de probabilités exacte. L'implémentation de cette méthode dans la librairie « quimb » [89] est utilisée dans ce mémoire pour réduire la complexité de l'échantillonnage.

Bibliographie

1. Shor, P. *Algorithms for Quantum Computation : Discrete Logarithms and Factoring* in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), 124-134. doi :10.1109/SFCS.1994.365700.
2. Cerezo, M. *et al.* Variational Quantum Algorithms. *Nature Reviews Physics* **3,** 625-644. doi :10.1038/s42254-021-00348-9 (2021).
3. Jerrum, M. R., Valiant, L. G. & Vazirani, V. V. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science* **43,** 169-188. doi :10.1016/0304-3975(86)90174-X (1986).
4. Hadfield, S. *et al.* From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz. *Algorithms* **12,** 34. doi :10.3390/a12020034 (2019).
5. Farhi, E., Goldstone, J. & Gutmann, S. *A Quantum Approximate Optimization Algorithm* 2014. doi :10.48550/arXiv.1411.4028. arXiv : 1411.4028 [quant-ph].
6. Bärtschi, A. & Eidenbenz, S. *Grover Mixers for QAOA : Shifting Complexity from Mixer Design to State Preparation* in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2020), 72-82. doi :10.1109/QCE49297.2020.00020.
7. Roth, D. On the Hardness of Approximate Reasoning. *Artificial Intelligence* **82,** 273-302. doi :10.1016/0004-3702(94)00092-1 (1996).
8. Sang, T., Bearne, P. & Kautz, H. *Performing Bayesian Inference by Weighted Model Counting* in *Proceedings of the 20th National Conference on Artificial Intelligence* **1** (2005), 475-481.
9. Abramson, B., Brown, J., Edwards, W., Murphy, A. & Winkler, R. L. Hailfinder : A Bayesian System for Forecasting Severe Weather. *International Journal of Forecasting* **12,** 57-71. doi :10.1016/0169-2070(95)00664-8 (1996).
10. Valiant, L. G. The Complexity of Enumeration and Reliability Problems. *SIAM Journal on Computing* **8,** 410-421. doi :10.1137/0208032 (1979).

11. Duenas-Osorio, L., Meel, K., Paredes, R. & Vardi, M. *Counting-Based Reliability Estimation for Power-Transmission Grids* in *Proceedings of the AAAI Conference on Artificial Intelligence* **31** (2017), 4488-4494. doi :10.1609/aaai.v31i1.11178.
12. Jerrum, M. & Sinclair, A. Polynomial-Time Approximation Algorithms for the Ising Model. *SIAM Journal on Computing* **22,** 1087-1116. doi :10.1137/0222066 (1993).
13. Baluta, T., Shen, S., Shinde, S., Meel, K. S. & Saxena, P. *Quantitative Verification of Neural Networks and Its Security Applications* in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019), 1249-1264. doi :10.1145/3319535.3354245.
14. Aaronson, S. & Arkhipov, A. *The Computational Complexity of Linear Optics* in *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing* (2011), 333-342. doi :10.1145/1993636.1993682.
15. Bouland, A., Fefferman, B., Nirkhe, C. & Vazirani, U. On the Complexity and Verification of Quantum Random Circuit Sampling. *Nature Physics* **15,** 159-163. doi :10.1038/s41567-018-0318-2 (2019).
16. Agrawal, M., Kayal, N. & Saxena, N. PRIMES Is in P. *Annals of Mathematics* **160,** 781-793. eprint : 3597229. https://www.jstor.org/stable/3597229 (2004).
17. Cobham, A. in *Logic, Methodology and Philosophy of Science* 24-30 (North-Holland Pub. Co., 1965).
18. Edmonds, J. Paths, Trees, and Flowers. *Canadian Journal of Mathematics* **17,** 449-467. doi :10.4153/CJM-1965-045-4 (1965).
19. Sipser, M. *Introduction to the Theory of Computation* 1re éd. (International Thomson Publishing, 1996).
20. Carlson, J. A., Jaffe, A., Wiles, A., Institute, C. M. & Society, A. M. *The Millennium Prize Problems* (American Mathematical Soc., 2006).
21. Impagliazzo, R. & Paturi, R. On the Complexity of *K*-SAT. *Journal of Computer and System Sciences* **62,** 367-375. doi :10.1006/jcss.2000.1727 (2001).
22. Valiant, L. G. The Complexity of Computing the Permanent. *Theoretical Computer Science* **8,** 189-201. doi :10.1016/0304-3975(79)90044-6 (1979).
23. Toda, S. PP Is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing* **20,** 865-877. doi :10.1137/0220053 (1991).
24. Cook, S. A. *The Complexity of Theorem-Proving Procedures* in *Proceedings of the Third Annual ACM Symposium on Theory of Computing* (1971), 151-158. doi :10.1145/800157.805047.
25. Levin, L. A. Universal Sequential Search Problems. *Problems of information transmission* **9,** 256-266 (1973).

26. Marques-Silva, J. *Practical Applications of Boolean Satisfiability* in *2008 9th International Workshop on Discrete Event Systems* (2008), 74-80. doi :10.1109/WODES.2008.4605925.
27. Krom, M. R. The Decision Problem for a Class of First-Order Formulas in Which All Disjunctions Are Binary. *Zeitschrift fur mathematische Logik und Grundlagen der Mathematik* **13,** 15-20. doi :10.1002/malq.19670130104 (1967).
28. Karp, R. M. *Reducibility among Combinatorial Problems* in *Proceedings of a Symposium on the Complexity of Computer Computations* (1972), 85-103. doi :10.1007/978-1-4684-2001-2_9.
29. Schaefer, T. J. *The Complexity of Satisfiability Problems* in *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing* (1978), 216-226. doi :10.1145/800133.804350.
30. Froleyks, N., Heule, M., Iser, M., Järvisalo, M. & Suda, M. SAT Competition 2020. *Artificial Intelligence* **301,** 103572. doi :10.1016/j.artint.2021.103572 (2021).
31. Vazirani, V. V. *Approximation Algorithms* doi :10.1007/978-3-662-04565-7 (Springer, 2003).
32. Lund, C. & Yannakakis, M. On the Hardness of Approximating Minimization Problems. *Journal of the ACM* **41,** 960-981. doi :10.1145/185675.306789 (1994).
33. Biere, A., Heule, M., van Maaren, H. & Walsh, T. *Handbook of Satisfiability* (IOS Press, 2009).
34. Davis, M., Logemann, G. & Loveland, D. A Machine Program for Theorem-Proving. *Communications of the ACM* **5,** 394-397. doi :10.1145/368273.368557 (1962).
35. Kourtis, S., Chamon, C., Mucciolo, E. & Ruckenstein, A. E. Fast Counting with Tensor Networks. *SciPost Physics* **7,** 060. doi :10.21468/SciPostPhys.7.5.060 (2019).
36. Dudek, J. M., Dueñas-Osorio, L. & Vardi, M. Y. *Efficient Contraction of Large Tensor Networks for Weighted Model Counting through Graph Decompositions* 2020. doi :10.48550/arXiv.1908.04381. arXiv : 1908.04381 [cs].
37. Dudek, J. M. & Vardi, M. Y. *Parallel Weighted Model Counting with Tensor Networks* 2021. doi :10.48550/arXiv.2006.15512. arXiv : 2006.15512 [cs].
38. Wahlström, M. *A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances* in *Parameterized and Exact Computation* (éd. Grohe, M. & Niedermeier, R.) (2008), 202-213. doi :10.1007/978-3-540-79723-4_19.
39. Dahllöf, V., Jonsson, P. & Wahlström, M. Counting Models for 2SAT and 3SAT Formulae. *Theoretical Computer Science* **332,** 265-291. doi :10.1016/j.tcs.2004.10.037 (2005).

40. Stockmeyer, L. *The Complexity of Approximate Counting* in *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing* (1983), 118-126. doi :10.1145/800061.808740.
41. Timme, M., van Bussel, F., Fliegner, D. & Stolzenberg, S. Counting Complex Disordered States by Efficient Pattern Matching : Chromatic Polynomials and Potts Partition Functions. *New Journal of Physics* **11,** 023001. doi :10.1088/1367-2630/11/2/023001 (2009).
42. Brassard, G., Høyer, P., Mosca, M. & Tapp, A. in *Quantum Computation and Information (Washington, DC, 2000)* 53-74 (Amer. Math. Soc., Providence, RI, 2002). doi :10.1090/conm/305/05215.
43. Wie, C.-R. Simpler Quantum Counting. *Quantum Information and Computation* **19,** 0967-0983. doi :10.26421/QIC19.11-12-5 (2019).
44. Aaronson, S. & Rall, P. in *2020 Symposium on Simplicity in Algorithms* 24-32 (Society for Industrial and Applied Mathematics, 2020). doi :10.1137/1.9781611976014.5.
45. Moore, C. & Mertens, S. *The Nature Of Computation* https://doi.org/10.1093/acprof:oso/9780199233212.001.0001 (Oxford University Press, 2011).
46. Mézard, M. & Montanari, A. *Information, Physics, and Computation* https://doi.org/10.1093/acprof:oso/9780198570837.001.0001 (Oxford University Press, 2009).
47. Achlioptas, D., Chtcherba, A., Istrate, G. & Moore, C. *The Phase Transition in 1-in-k SAT and NAE 3-SAT* in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms* (2001), 721-722.
48. Raymond, J., Sportiello, A. & Zdeborová, L. Phase Diagram of the 1-in-3 Satisfiability Problem. *Physical Review E* **76,** 011101. doi :10.1103/PhysRevE.76.011101 (2007).
49. Dyer, M. & Greenhill, C. On Markov Chains for Independent Sets. *Journal of Algorithms* **35,** 17-49. doi :10.1006/jagm.1999.1071 (2000).
50. Barthel, W. & Hartmann, A. K. Clustering Analysis of the Ground-State Structure of the Vertex-Cover Problem. *Physical Review E* **70,** 066120. doi :10.1103/PhysRevE.70.066120 (2004).
51. Zdeborová, L. *Statistical Physics of Hard Optimization Problems* 2008. doi :10.48550/arXiv.0806.4112. arXiv : 0806.4112 [cond-mat].
52. Trakhtenbrot, B. A. *On Autoreducibility* in *Doklady Akademii Nauk* **192** (1970), 1224-1227.
53. Selke, J. *Autoreducibility and Friends : About Measuring Redundancy in Sets* thèse de doct. (Universität Hanover, 2006).

54. Hemaspaandra, L. A. in *Complexity and Approximation : In Memory of Ker-I Ko* 19-47 (Springer International Publishing, 2020). doi :10.1007/978-3-030-41672-0_3.
55. Schnorr, C.-P. Optimal Algorithms for Self-Reducible Problems. *Proccedings of the 3rd International Colloquium on Automata, Languages and Programming* **3,** 322-337 (1976).
56. Goldreich, O. Computational Complexity : A Conceptual Perspective. *SIGACT News* **39,** 35-39. doi :10.1145/1412700.1412710 (2008).
57. Khuller, S. & Vazirani, V. V. Planar Graph Coloring Is Not Self-Reducible, Assuming P≠NP. *Theoretical Computer Science* **88,** 183. doi :10.1016/0304-3975(91)90081-C (1991).
58. Jerrum, M. & Sinclair, A. Fast Uniform Generation of Regular Graphs. *Theoretical Computer Science* **73,** 91-100. doi :10.1016/0304-3975(90)90164-D (1990).
59. Chakraborty, S., Meel, K. S. & Vardi, M. Y. *A Scalable and Nearly Uniform Generator of SAT Witnesses* in *Computer Aided Verification* (Springer, 2013), 608-623. doi :10.1007/978-3-642-39799-8_40.
60. Jerrum, M. *Counting, Sampling and Integrating : Algorithm and Complexity* doi :10.1007/978-3-0348-8005-3 (Birkhäuser Basel, 2003).
61. Broder, A. Z. *How Hard Is It to Marry at Random ?* (*On the Approximation of the Permanent*) in *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing* (1986), 50-58. doi :10.1145/12130.12136.
62. Sinclair, A. *Algorithms for Random Generation and Counting : A Markov Chain Approach* doi :10.1007/978-1-4612-0323-0 (Birkhäuser, 1993).
63. Born, M. & Fock, V. Beweis des Adiabatensatzes. *Zeitschrift für Physik* **51,** 165-180. doi :10.1007/BF01343193 (1928).
64. Albash, T. & Lidar, D. A. Adiabatic Quantum Computation. *Reviews of Modern Physics* **90,** 015002. doi :10.1103/RevModPhys.90.015002 (2018).
65. Messiah, A. *Quantum Mechanics* (North-Holland Publishing Company, 1961).
66. Amin, M. H. S. Consistency of the Adiabatic Theorem. *Physical Review Letters* **102,** 220401. doi :10.1103/PhysRevLett.102.220401 (2009).
67. Farhi, E., Goldstone, J., Gutmann, S. & Sipser, M. *Quantum Computation by Adiabatic Evolution* 2000. doi :10.48550/arXiv.quant-ph/0001106. arXiv : quant-ph/0001106.
68. Altshuler, B., Krovi, H. & Roland, J. Anderson Localization Makes Adiabatic Quantum Optimization Fail. *Proceedings of the National Academy of Sciences* **107,** 12446-12450. doi :10.1073/pnas.1002116107 (2010).

69. Nishimori, H. & Takada, K. Exponential Enhancement of the Efficiency of Quantum Annealing by Non-Stoquastic Hamiltonians. *Frontiers in ICT* **4.** doi :10.3389/fict.2017.00002 (2017).
70. Hormozi, L., Brown, E. W., Carleo, G. & Troyer, M. Nonstoquastic Hamiltonians and Quantum Annealing of an Ising Spin Glass. *Physical Review B* **95,** 184416. doi :10.1103/PhysRevB.95.184416 (2017).
71. Wurtz, J. & Love, P. J. Counterdiabaticity and the Quantum Approximate Optimization Algorithm. *Quantum* **6,** 635. doi :10.22331/q-2022-01-27-635 (2022).
72. Zhou, L., Wang, S.-T., Choi, S., Pichler, H. & Lukin, M. D. Quantum Approximate Optimization Algorithm : Performance, Mechanism, and Implementation on Near-Term Devices. *Physical Review X* **10,** 021067. doi :10.1103/PhysRevX.10.021067 (2020).
73. Blekos, K. *et al.* A Review on Quantum Approximate Optimization Algorithm and Its Variants. *Physics Reports. A Review on Quantum Approximate Optimization Algorithm and Its Variants* **1068,** 1-66. doi :10.1016/j.physrep.2024.03.002 (2024).
74. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information : 10th Anniversary Edition* (Cambridge University Press, 2011).
75. Farhi, E., Gamarnik, D. & Gutmann, S. *The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph : A Typical Case* 2020. doi :10.48550/arXiv.2004.09002. arXiv : 2004.09002 [quant-ph].
76. Egger, D. J., Mareček, J. & Woerner, S. Warm-Starting Quantum Optimization. *Quantum* **5,** 479. doi :10.22331/q-2021-06-17-479 (2021).
77. Lucas, A. Ising Formulations of Many NP Problems. *Frontiers in Physics* **2.** doi :10.3389/fphy.2014.00005 (2014).
78. Lodewijks, B. *Mapping NP-hard and NP-complete Optimisation Problems to Quadratic Unconstrained Binary Optimisation Problems* 2020. doi :10.48550/arXiv.1911.08043. arXiv : 1911.08043 [cs].
79. Sack, S. H. & Serbyn, M. Quantum Annealing Initialization of the Quantum Approximate Optimization Algorithm. *Quantum* **5,** 491. doi :10.22331/q-2021-07-01-491 (2021).
80. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren Plateaus in Quantum Neural Network Training Landscapes. *Nature Communications* **9,** 4812. doi :10.1038/s41467-018-07090-4 (2018).
81. Larocca, M. *et al. A Review of Barren Plateaus in Variational Quantum Computing* 2024. doi :10.48550/arXiv.2405.00781. arXiv : 2405.00781 [quant-ph].

82. Anschuetz, E. R. & Kiani, B. T. Quantum Variational Algorithms Are Swamped with Traps. *Nature Communications* **13,** 7760. doi :10.1038/s41467-022-35364-5 (2022).

83. Bittel, L. & Kliesch, M. Training Variational Quantum Algorithms Is NP-Hard. *Physical Review Letters* **127,** 120502. doi :10.1103/PhysRevLett.127.120502 (2021).

84. Matsuda, Y., Nishimori, H. & Katzgraber, H. G. Ground-State Statistics from Annealing Algorithms : Quantum versus Classical Approaches. *New Journal of Physics* **11,** 073021. doi :10.1088/1367-2630/11/7/073021 (2009).

85. Mandrà, S., Zhu, Z. & Katzgraber, H. G. Exponentially Biased Ground-State Sampling of Quantum Annealing Machines with Transverse-Field Driving Hamiltonians. *Physical Review Letters* **118,** 070502. doi :10.1103/PhysRevLett.118.070502 (2017).

86. Sundar, B., Paredes, R., Damanik, D. T., Dueñas-Osorio, L. & Hazzard, K. R. A. *A Quantum Algorithm to Count Weighted Ground States of Classical Spin Hamiltonians* 2019. doi :10.48550/arXiv.1908.01745. arXiv : 1908.01745 [quant-ph].

87. Xie, N. *et al.* Performance Upper Bound of a Grover-mixer Quantum Alternating Operator Ansatz. *Physical Review A* **111,** 012401. doi :10.1103/PhysRevA.111.012401 (2025).

88. Viger, F. & Latapy, M. *Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence* in *Computing and Combinatorics* (2005), 440-449. doi :10.1007/11533719_45.

89. Gray, J. Quimb : A Python Package for Quantum Information and Many-Body Calculations. *Journal of Open Source Software* **3,** 819. doi :10.21105/joss.00819 (2018).

90. Virtanen, P. *et al.* SciPy 1.0 : Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17,** 261-272. doi :10.1038/s41592-019-0686-2 (2020).

91. Sharma, S., Roy, S., Soos, M. & Meel, K. S. *GANAK : A Scalable Probabilistic Exact Model Counter* in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (2019), 1169-1176. doi :10.24963/ijcai.2019/163.

92. Eén, N. & Sörensson, N. *An Extensible SAT-solver* in *Theory and Applications of Satisfiability Testing* (2004), 502-518. doi :10.1007/978-3-540-24605-3_37.

93. Audemard, G. & Simon, L. *Predicting Learnt Clauses Quality in Modern SAT Solvers* in *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (2009), 399-404.

94. Ignatiev, A., Morgado, A. & Marques-Silva, J. *PySAT : A Python Toolkit for Prototyping with SAT Oracles* in *Theory and Applications of Satisfiability Testing – SAT 2018* (2018), 428-437. doi :10.1007/978-3-319-94144-8_26.

95. Ochoa, A. J., Jacob, D. C., Mandrà, S. & Katzgraber, H. G. Feeding the Multitude : A Polynomial-Time Algorithm to Improve Sampling. *Physical Review E* **99,** 043306. doi :10.1103/PhysRevE.99.043306 (2019).

96. Gomes, C. P., Hoffmann, J., Sabharwal, A. & Selman, B. *From Sampling to Model Counting* in *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (2007), 2293-2299.

97. Bridgeman, J. C. & Chubb, C. T. Hand-Waving and Interpretive Dance : An Introductory Course on Tensor Networks. *Journal of Physics A : Mathematical and Theoretical* **50,** 223001. doi :10.1088/1751-8121/aa6dc3 (2017).

98. Biamonte, J. & Bergholm, V. *Tensor Networks in a Nutshell* 2017. doi :10.48550/arXiv.1708.00006. arXiv : 1708.00006 [quant-ph].

99. Baker, T. E., Desrosiers, S., Tremblay, M. & Thompson, M. P. Méthodes de Calcul Avec Réseaux de Tenseurs En Physique. *Canadian Journal of Physics* **99,** 207-221. doi :10.1139/cjp-2019-0611 (2021).

100. Gray, J. & Kourtis, S. Hyper-Optimized Tensor Network Contraction. *Quantum* **5,** 410. doi :10.22331/q-2021-03-15-410 (2021).

101. Ferris, A. J. & Vidal, G. Perfect Sampling with Unitary Tensor Networks. *Physical Review B* **85,** 165146. doi :10.1103/PhysRevB.85.165146 (2012).