Projet #1 - Rapport final

BSQ 101 - Projets intégrateurs en programmation quantique

Algorithme de Grover

Presenté par

Henri-Louis Charbonneau
Wilhem Harvey
Sahar Saoudi

Table des matières

1 | Introduction

L'algorithme de Grover, conçu par Lov Grover en 1996[1], est un pilier essentiel de l'informatique quantique. Il propose une solution à la recherche non structurée, permettant une accélération quadratique par rapport aux algorithmes classiques lors de la recherche dans une base de données non triée[3]. Grâce à des principes de la mécanique quantique comme la superposition et l'interférence, l'application de cet algorithme permet d'augmenter les probabilités d'obtenir un résultat désiré lorsqu'on mesure un système quantique[2]. Cette propriété vaut à l'algorithme plusieurs applications pratiques significatives, notamment dans les domaines de la cryptographie, de la recherche de données et de l'optimisation. Pour donner un exemple concret de l'utilité de l'algorithme en question dans un contexte de recherche de données, supposons un ensemble de X personnes. Un individu désire trouver quelqu'un dans cet ensemble sachant cuisiner son mets préféré, une crème de brocoli. Sur X personnes, Y sont capables de réaliser cette tâche. Si ledit individu tente de trouver au hasard une personne pouvant lui cuisiner une crème de brocoli, celui-ci aura une chance de trouver une telle personne. Si Y est significativement plus petit que X, un temps important devra être investi afin de trouver une personne Y. L'algorithme de Grover permettra alors d'amplifier les chances que l'individu trouve au hasard une personne Y capable de lui cuisiner une crème de brocoli. De plus, l'algorithme de Grover se distingue particulièrement par son aptitude à résoudre des problèmes de satisfiabilité (SAT)[1]. Ce type de problème implique un ensemble de variables booléennes (vrai ou faux), liées par des opérateurs logiques (et, ou, non) pour générer des propositions logiques. L'algorithme est composé de deux sections principales, l'oracle et le diffuseur, qui modifient les états quantiques et permettent d'obtenir les résultats énoncés plus haut. Ainsi, l'algorithme de Grover est aussi utile que son fonctionnement est intéressant.

2 | Cadre théorique

Dans cette section, le fonctionnement de l'algorithme de Grover sera expliqué de façon théorique. Tout d'abord, il est important de comprendre que l'implémentation de l'algorithme de Grover créé pour ce projet a pour but de traiter des formules logiques sous forme normal conjonctive (des conjonctions de disjonctions) et utilise les variables et les clauses de cette formule pour traiter les problèmes. L'algorithme de Grover débute par l'application de portes Hadamard sur tous les qubits représentants les variables d'une formule logique afin de préparer un état de superposition. Ainsi, l'état initial des qubits peut être décrit de cette façon :

$$|0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

Où $|0\rangle$ est l'état initial des qubits, n est le nombre de qubits et $|x\rangle$ représente les états de bases possibles pour n qubits. La première section importante de l'algorithme est l'oracle, qui est responsable d'inverser la phase des états que l'on désire obtenir et laisse la phase des états qui ne sont pas intéressant inchangée. Pour ce faire, les qubits représentant les variables d'une formule logique seront intriqués avec les qubits représentant les clauses de cette même formule. Une porte z multi contrôlée sera alors appliquée aux qubits des clauses, ce qui, dû à l'intrication, aura l'effet décrit plus tôt. Ainsi, si l'on applique l'oracle à un état $|x\rangle$ désiré, le résultat suivant sera obtenu :

$$U_{\text{oracle}}|x\rangle = -|x\rangle$$

Dans le cas contraire, si l'état $|x\rangle$ auquel l'oracle est appliqué n'est pas intéressant, le résultat suivant sera obtenu :

$$U_{\text{oracle}}|x\rangle = |x\rangle$$

Le fonctionnement du diffuseur sera maintenant décrit. Lorsqu'on l'applique à un état dont la phase a été inversée par l'oracle, le rôle du diffuseur est de faire effectuer à cet état une réflexion autour de l'état de superposition dans l'espace des états. Ainsi, ce procédé amplifiera les probabilités de mesurer les états recherchés par rapport aux états non recherchés. Le procédé en question appliqué à l'état $|\phi\rangle$ peut-être exprimé de la façon suivante :

$$U_{\text{diffuseur}}|\phi\rangle = |\phi\rangle \quad \text{si } |\phi\rangle \text{ est égale à l'état de superposition dans l'espace des états}$$

$$U_{\text{diffuseur}}|\phi\rangle = -|\phi\rangle \quad \text{si } |\phi\rangle \text{ est orthogonale à l'état de superposition dans l'espace des états}$$

L'application de l'oracle et du diffuseur représente donc une itération de l'algorithme de Grover. Il est possible de répéter ce processus pour amplifier davantage les probabilités de mesurer des états recherchés. Cependant, il est important de prendre en compte que trop d'application de l'algorithme de Grover aurait l'effet inverse de ce qui est désiré, c'est-à-dire diminuer la probabilité de mesurer des états recherchés.

3 | Implémentation de l'algorithme

L'algorithme de Grover implémenté pour ce projet comprend 4 fonctions principales regroupées dans 2 fichiers. Les fonctions build_grover_circuit et solve_sat_with_grover se trouvent dans le fichier grover_build.py. Les fonctions cnf_to_oracle et build_diffuser (ainsi que 2 sous-fonctions) se trouvent dans le fichier grover_utils.py.

3.1 | build_grover_circuit

Cette fonction est responsable de construire le circuit complet de l'algorithme. Elle prend en entrée le circuit de l'oracle, le nombre de variables dans la formule logique et le nombre d'itération de l'algorithme à appliquer et retourne le circuit correspondant à l'entièreté de l'Algorithme de Grover à la fin du processus. Lors de l'exécution de cette fonction, deux registres quantiques et un registre classique sont créés. Le premier registre quantique correspond à chaque variable de la formule logique et le second représente les clauses comprises dans la formule logique traitée. Quant à lui, le registre classique sera chargé de faire les mesures des qubits associés à chaque variable après à la fin de l'exécution de l'algorithme. Le circuit est ensuite généré. Chaque qubit des variables se verra appliquer une porte Hadamard pour atteindre une superposition d'état. Le diffuseur sera par la suite construit par un appel de la fonction build_diffuser. La dernière partie de cette fonction consiste à ajouter l'oracle et le diffuseur au circuit le nombre de fois indiqué en entrée. Le circuit est ensuite retourné.

3.2 | solve_sat_with_grover

Cette fonction calcule et retourne les résultats obtenus lorsqu'on applique l'algorithme à une formule logique. Elle prend en entrée une formule logique, la fonction générant l'oracle et un simulateur et retourne un ou plusieurs dictionnaires correspondant aux solutions de la formule logique. Au début de la fonction. Le circuit de Grover est généré avec un appel de la fonction précédemment décrite. Les qubits représentant les variables de la formule logique sont ensuite mesurés. Le circuit est alors exécuté avec le simulateur spécifié en entrée et le nombre de coups pour obtenir les résultats. La suite de la fonction est responsable de créer les dictionnaires contenant les réponses. La moyenne des mesures sur chaque qubit des variables est calculée et ensuite utilisée comme référence. Si une variable est mesurée au moins deux fois plus que la moyenne de mesure de toutes les variables, ladite variable sera alors désignée comme une solution valable et se fera attribuer la valeur booléenne True dans le dictionnaire. Dans le cas contraire, False sera attribué à la variable. Le ou les dictionnaires représentant chaque solution possible seront par la suite retournés.

3.3 | cnf_to_oracle

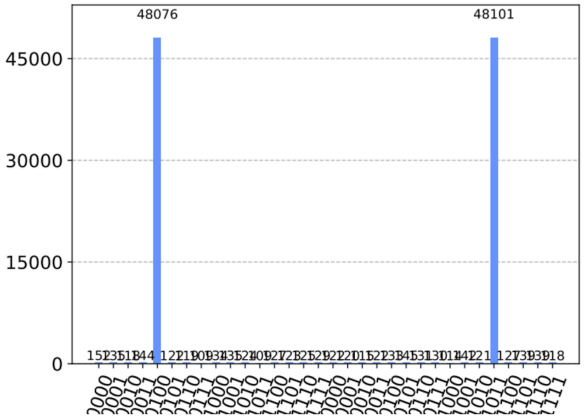
Cette fonction construit le circuit de l'oracle. Elle prend en entrée une formule logique et retourne le circuit de l'oracle. Les variables de la formule logique sont d'abord mises dans un ordre fixe permettant des manipulations plus simples par la suite. Deux registres quantiques sont créés pour cette fonction : le registre des variables et le registre des clauses. Le circuit est ensuite généré. Une boucle itère pour chacune des clauses et crée une porte X multicontrôlée correspondant à chaque clause en utilisant la sous-fonction createMCXForClause. Des portes X sont ensuite appliquées à chaque qubit des clauses pour respecter la règle de De Morgan. À ce stade, les qubits des variables et les qubits des clauses sont intriqués. Une copie du circuit construit jusqu'à maintenant est faite et inversée pour simplifier l'implémentation. Une porte Z multi contrôlée est appliquée sur tous les qubits des clauses avec la sous fonction createMCZGate. Cela aura pour effet d'inverser la phase des qubits des variables ayant des états qui sont recherchés. Par la suite, la copie du circuit inversée est appliquée sur le circuit pour ainsi désintriquer les deux registres de qubits. Le circuit final de l'oracle est ensuite retourné.

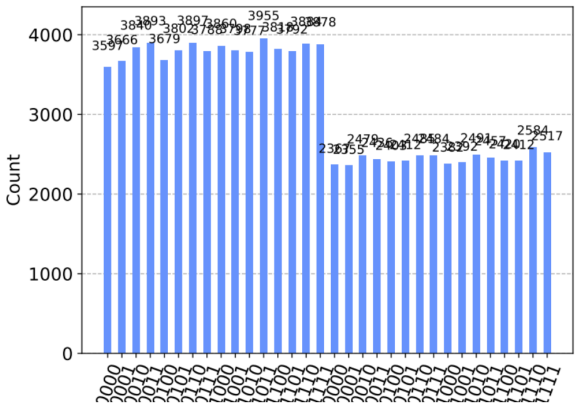
3.4 | build_diffuser

Cette fonction construit le circuit du diffuseur. Elle prend en entrée le nombre de variables dans la formule logique et retourne le circuit du diffuseur. Le registre des variables est le seul registre quantique créé dans cette fonction. Le circuit est ensuite créé. Chaque qubit se voit appliquer une porte Hadamard pour atteindre une superposition d'état. Ensuite des portes X ainsi qu'une porte Z multicontrôlée (créé par la sous-fonction createMCZGate) sont appliquées à chacun des qubits pour obtenir essentiellement une porte Z multicontrôlée par l'état zéro. Ainsi, la probabilité de mesurer les états des qubits dont la phase a été inversée par l'oracle sera amplifiée. Des portes X et des portes Hadamard sont ensuite respectivement appliquées pour effacer les changements amenés par l'application de ces mêmes portes plus tôt. Le circuit du diffuseur est ensuite retourné.

4 | Résultats

Durant ce projet, l'algorithme développé fut notamment appliqué à deux problématiques différentes : le problème du gâteau et le problème de la planète Pincus. Pour ce rapport, les résultats obtenus pour la problématique du gâteau seront analysés en détail. Le problème du gâteau consiste à trouver à l'aide de différents énoncés logiques qui a mangé la dernière part de gâteau. Voici les résultats obtenus après l'application de 3 itérations de l'algorithme de Grover à cent mille coups à cette problématique :





5 | Analyse des résultats et avantages de la solution quantique

Lorsqu'on analyse les résultats de la problématique du gâteau, il est possible d'observer avec les résultats du simulateur Aer que deux réponses sont possibles : soit Chris à manger la part de gâteau (00100), soit tout le monde sauf Chris a mangé la part de gâteau (11011). Les résultats obtenus lorsqu'on applique Grover à la problématique du gâteau sur un ordinateur quantique sont moins concluants. En effet, après 3 itérations de l'algorithme et cent mille coups, ils semblent que les états où Emma n'a pas manger le gâteau sont les plus probables, mais cela ne permet pas de conclure sur une ou plusieurs solutions évidentes. Deux réponses sont également possibles pour la problématique de la planète Pincus lorsqu'on effectue un test sur le simulateur Aer : soit les habitants sont courageux, heureux, malades et bruyants ou soit ils sont courageux, heureux, en santé et bruyants. Leur état de santé est donc inconnu. Ainsi, l'utilisation d'une solution quantique pour ce genre de problème augmente drastiquement les chances de mesurer au hasard un état correspondant à un critère précédemment établi. Si l'on analyse les résultats obtenus lorsqu'on applique 3 itérations de l'algorithme de Grover à cent mille coups à la problématique du gâteau (avec le simulateur Aer), on observe qu'un état recherché a été mesuré $\frac{96177}{100000}$ ($\approx 0,96$) coups. Cette probabilité est nettement supérieure à $\frac{2}{32}$ ($\approx 0,063$), la probabilité de trouver un état désiré au hasard. De plus, trouver un élément précis dans un ensemble à une complexité algorithmique de O(n). L'utilisation de l'algorithme de Grover réduit cette complexité à $O(\sqrt{n})$ [4]. Il est donc possible de conclure que l'utilisation de solution quantique, plus précisément de l'algorithme de Grover, offre un avantage non négligeable à une solution classique pour ce type de problème.

6 | References

[1] Algorithme de grover. Consulté le 20 janvier 2024.

[2] Maxime Dion. Algorithmes quantiques, 2024.

[3] Nicolas Macris. Algorithme de grover, 2012.

[4] tedhudek dphansen geduardo dime10 SoniaLopezBravo, Bradben. Théorie de l'algorithme de recherche de grover, 2023.