Algorithme de Grover

Application à la résolution de problèmes de satisfiabilité

Thomas Alexandre, Camilo Fernandez, et Louis-Félix Vigneux

29 janvier 2024

1 Introduction

L'algorithme de Grover permet de résoudre un problème de satisfiabilité. Ce type de problème, surnommés SAT, se pose comme un ensemble de propositions vérifiables par des variables booléennes. Le problème est dit satisfiable si on peut attribuer une valeur (Vrai ou Faux) à chacune des variables booléennes. Autrement dit, le problème possède une solution valide. Dans ce rapport, on résout un problème SAT dans lequel des invités sont coupables (Vrai) ou non (Faux) d'avoir manger notre part de gâteau.

Une application concrète de ce type de problème est de trouver un item dans une base de données désordonnée de manière efficace. En effet, en utilisant la superposition et l'intrication, deux propriétés employées par l'algorithme de Grover, il serait vraiment plus intéressant d'employer cet algorithme pour chercher un numéro de téléphone dans un botin. Le temps sauvé comparé à un ordinateur classique est significatif. En considérant les éléments de la base de données comme des vecteurs, il est possible d'isoler les vecteurs des éléments désirés, $|w\rangle$. Par la suite, nous allons inverser la phase de ce vecteur pour ensuite le réfléchir par rapport au vecteur uniforme. Ainsi, après plusieurs cycles d'inversion de phase et de réflexion, nous améliorons drastiquement les probabilités d'obtenir le bon résultat.

2 Notions théorique essentielles

L'algorithme de Grover utilise la propriété de superposition afin de compléter plusieurs opérations en même temps. Cela apporte une amélioration au temps d'exécution des algorithmes classiques qui s'opèrent séquentiellement. En effet, cela revient à la définition de parrallélisme quantique. La comparaison asymptotique de l'algorithme classique de recherche tend vers O(N) évaluations nécessaires tant dis que l'algorithme quantique tend vers O($\sqrt{N}$) évaluations ($N = 2^n$ où n est le nombre de variables dans le problème de satisfiabilité). Cette réduction exponentielle du nombre d'évaluation d'une banque de données présente un avantage significatif de l'algorithme de Grover en termes de durée de l'exécution.

Oracle

L'oracle a pour objectif de modifier la phase des états-solution. D'abord, il faut des qubits pour vérifier la proposition logique (associés à la valeur de vérité des variables) et des qubits ancillaires (chacun d'eux sont associés à une clause de la proposition logique). Ainsi, en superposant tous les qubits propositionnels avant l'oracle, on obtient des états intriqués entre les qubits propositionnels et les qubits ancillaires. Les états-solutions sont intriqués selon la forme $|1\rangle^n \, ancillaires \otimes |\psi\rangle$ où les qubits ancillaires sont dans l'état $|1\rangle$ lorsque le circuit représentant la proposition logique est appliqué.

Pour ce qui est du circuit représentant l'oracle, on applique d'abord la formule logique traduite en porte quantique contrôlée. Effectivement, chaque close est controlée par l'état des variables propositionnelles impliquées. Ensuite, on applique la porte contrôle $\hat{Z}$ sur chacun des qubits ancillaires. Si toute les clauses sont repectées (tout les qubits ancilaires sont dans l'état $|1\rangle$) on inverse la phase de cet état. Finalement, on applique à nouveau la formule logique traduite afin de remettre l'état des qubits ancillaire à 0. De ce fait, seulement les états validant la proposition logique voient leur phase se faire inverser et les qubits ancilaires sont désintriqués de l'état des variables propositionnelles. Cela permet d'appliquer le circuit diffuseur sans effet adverse, puis de réitérer l'algorithme de Grover si nécessaire.

Diffuseur

Le diffuseur a pour objectif d'amplifier la phase des états-solutions. Le circuit du diffuseur permet de faire une réflexion de la phase de l'état par rapport à l'axe de superposition uniforme. Pour réussir cet effet, le circuit du diffuseur est construit en appliquant des portes $\hat{H}$ et $\hat{X}$ sur chacun des qubits propositionnels. Après cela, une porte contrôle $\hat{Z}$ est appliquée sur l'entièreté des qubits propositionnels. Finalement, nous réappliquons les portes $\hat{X}$ et $\hat{H}$. Puisque l'état des bons qubits est inversé avec l'oracle, la réflexion de ces derniers par rapport à l'axe de superposition uniforme augmentera leur probabilité

d'être mesuré. En effet, nous pouvons nous imaginer cette augmentation avec un cercle où l'axe horizontal est représenté par le vecteur des mauvais états et l'axe vertical par celui des bons états. En faisant d'abbord une inversion de phase avec l'oracle, la réflexion avec l'axe de superposition uniforme sera de plus en plus grande. Ainsi, après plusieurs itérations. le vecteur analysé se rapprochera grandement de l'axe vertical, et par le fait même, des états solutions. En effet, selon ces axes, la probabilité de mesurer un bon résulat est identifié par le carré du sinus de l'angle du vecteur par rapport à l'axe horizontal. En augmentant cet angle, la valeur du sinus croit aussi. Nous avons ainsi une plus grande probabilité de mesurer un état-solution qu'à la situation initiale.

3 Implémentation

Oracle

Cette fonction transforme une formule logique de forme normale conjonctive en circuit quantique afin d'appliquer un changement de phase aux états-solution du problème de satisfiabilité. D'abord, il faut appliquer le circuit associé à la formule logique conjonctive. Cette partie se fait grâce à deux sous-fonctions décrites au prochain paragraphe. Par la suite, on modifie la phase des bons états par une porte C$\hat{Z}$ appliquée à tous les qubits ancillaires. Ensuite, on inverse le circuit représentant la formule logique pour désintriquer les qubits variables et les qubits ancillaires.

La sous-fonction transformant la formule logique en circuit quantique analyse chacune des clauses de la proposition. Chacune de celle-ci correspond à une porte multi-C$\hat{X}$. Ainsi, les variables des arguments agissent en tant que qubits contrôles où les variables associées à un complément s'activent en $|1\rangle$ et sinon, elles s'activent en $|0\rangle$. Ces états contrôles sont déterminés pour une autre sous-fonction. Donc, les arguments valides transforment l'état d'un qubit ancillaire à $|1\rangle$. Par la suite, puisque chacune des clauses sont une combinaison de disjonctions, nous appliquons une porte $\hat{X}$ à chacun des qubits ancillaires pour respecter la loi de De Morgan. Puisque chacun de ce ces qubits représentent une clause, s'ils sont tous dans l'état $|1\rangle$, alors l'état en entrée est une solution du problème de satisfiabilité.

Diffuseur

L'implémentation du diffuseur est assez directe. En effet, en prenant le nombre de qubits propositionnels en entrée, il suffit d'appliquer sur ceux-ci les portes qui ont été décrites dans la section précédente.

Fonction analysant les résultats

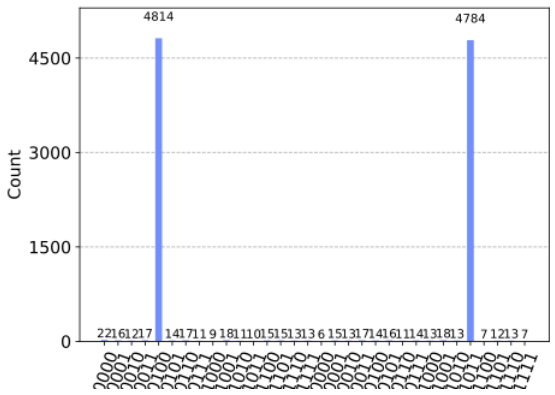
Nous avons aussi implémenté une fonction qui va exécuter l'algorithme de Grover tout en identifiant les résultats significatif. En effet, il exécute l'algorithme selon le nombre idéal d'itération qui est de $\pi/4 \cdot \sqrt{2^n}$ où n est le nombre de variables impliquées dans la proposition. Par la suite, solve_sat_with_grover utilise une fonction qui va retourner tous les états ayant des *counts* supérieurs à un écart-type au-dessus de la moyenne de la fréquence de tous les autres résultats. Par la suite, celle-ci en appelle une autre fonction qui transforme ces données dans un format plus facile à interpréter. En effet, cette sous-fonction est aussi le retour de la fonction solve_sat_with_grover qui est une liste de dictionnaire associant la valeur de vérité de chacune des variables pour l'ensemble des bonnes combinaisons.

4 Résultats

Pour cette section, nous allons nous concentrer sur l'application de notre algorithme sur une mise en situation : celle du gâteau. Il est important de noter que nous tirons les mêmes conclusions aussi avec une formule logique inventée et celle de la planète Pincus.
Donc, nous avons testé nos résultats sur le problème du gâteau avec quatre itérations de l'algorithme. En analysant la formule conjonctive suivante.

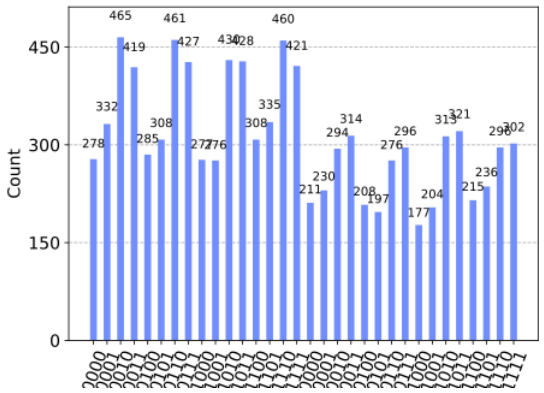
$$((\neg e \wedge \neg b) \vee (e \wedge b)) \wedge ((\neg c \wedge e) \vee (c \wedge \neg e)) \wedge ((e \wedge a) \vee (\neg e \wedge \neg a)) \wedge ((c \wedge \neg b) \vee (\neg c \wedge b)) \wedge ((d \wedge a) \vee (\neg d \wedge \neg a))$$

où a représente Alan, b représente Ben, c représente Chris, d représente Dave et e représente Emma. La variable est vraie si la personne a mangé le gateau.
Nous pouvons ensuite modifier cette équation en forme conjonctive normale pour l'inclure dans notre algorithme. Voici les résultats de l'algorithme :
Sur le simulateur



Résultats : [a : False, b : False, c : True, d : False, e : False, a : True, b : True, c : False, d : True, e : True]. Les résultats sont tous vrais.

Sur l'ordinateur quantique



Résultats : [a : False, b : True, c : True, d : False, e : False, a : True, b : True, c : False, d : False, e : False, a : True, b : True, c : True, d : False, e : False, a : False, b : True, c : True, d : True, e : False, a : True, b : True, c : True, d : True, e : False, a : True, b : True, c : False, d : True, e : False, a : False, b : True, c : False, d : False, e : False, a : False, b : True, c : False, d : True, e : False]. Les résultats ne sont pas tous vrais

Analyse des résultats

Nous voyons que nous avons les bons résultats avec le simulateur avec de très bonnes probabilités. En effet, avec la fonction subs de *sympy*, nous pouvons conclure avec succès que nous arrivons aux deux bonnes combinaisons de variable. Par contre, nous ne pouvons pas en tirer les même conclusions lorsque nous roulons l'algorithme sur un ordinateur. En effet, nous avons plusieurs combinaisons de variables significatives en retour. De plus, les bons "états" ne sont pas les plus amplifiés. Ainsi, le bruit quantique a un grand impact sur nos résultats.

Si nous nous fions qu'au simulateur, nous remarquons que l'algorithme de Grover présente un réel avantage quantique. Effectivement, nous remarquons clairement que 2 résultats sont ressortis du lot et ils sont véridiques comme mentionné plus tôt. Si nous utilisons une approche classique pour obtenir une bonne combinaison, nous avons besoin d'en moyenne 16 tentatives ($\frac{32}{2}$). En analysant les résultats de l'algorithme de Grover, nous obtenons les bons résulats 9 598 fois sur 10 000 tentatives. En utilisant la même logique que précédement, nous remarquons que seulement 1,04 tentatives en moyenne sont nécessaires pour trouver une combinaison de variable valide avec l'algorithme. En utilisant notre circuit, nous avons donc en moyenne, près de 16 fois plus de chances d'obtenir une bonne combinaison de valeurs de vérité. Nous avons alors un réel avantage quantique. En effet, chercher un élément dans une grande base de donnée sera encore plus efficace avec les ordinateurs quantique. Cette conclusion est confirmée aussi par ce qui a été mentionné dans la section 2 en comparant les deux complexité algorithmiques. Ainsi, l'algorithme de Grover présente un réel avantage quantique.

Sources :

DION, Maxime «Algorithmes quantiques». *Université de Sherbrooke*, (Consulté le 28 janvier 2023).

MATUSCHAK, Andy et NIELSEN Micheal «How the quantum serach algorithm works». *Quantum county*, [En ligne], [https ://quantum.country/search], (Consulté le 28 janvier 2023).