Projet : résolution quantique du sac à dos

UE Informatique quantique et Recherche opérationnelle

7 janvier 2025

ensIIE - MPRO

Ce TP noté se fera en trinôme, et devra être restitué (code + réponses aux questions) en au plus tard **le 19 janvier 2025 à 23h59** sur `exam.ensiie.fr` dans le dépôt `iqro-tp-note`. Le temps imparti encadré est de trois fois 1h45, soit 4h45 au total.

Il est conseillé pour les personnes n'ayant pas suivi de cours de recherche opérationnelle de se mettre en binôme avec une personne ayant suivi un tel cours.

Le but de ce TP est de résoudre le problème de sac à dos : QAOA et l'algorithme de Grover.

Les codes devront être rendus dans des fichiers `.py`; tous dans un unique dossier `code`. Des fichiers jupyter ou équivalents ne sont pas attendus. Les questions théoriques doivent être rendues dans un dossier `questions` dans un ou plusieurs fichiers pdf (latex compilé ou odt exporté) ou des images scannées d'une page manuscrite (de préférence de taille maîtrisée). Indiquer dans le code et les rapports le numéro de la question à laquelle vous répondez.

1 Description du problème et question préliminaire

Soit un sac de volume V et un ensemble d'objets de volumes respectivement $v_1, v_2, \ldots, v_n$ et d'utilités $u_1, u_2, \ldots, u_n$. On veut placer dans le sac une partie des objets de sorte que le sac soit plein et que la somme des utilités soit maximum. On ne peut pas découper les objets ni faire dépasser des objets du sac.

1. Écrire le modèle mathématique de ce problème, où les variables de décision sont binaires.

2. Ecrire une classe Knapsack qui représente une entrée du problème de sac à dos

3. Ecrire une fonction qui prend en entrée une instance du problème de sac à dos et résout l'instance. La réponse de la fonction doit être la solution et son utilité.

   Vous pouvez utiliser un algorithme bruteforce ou un algorithme de programmation dynamique, le second vous permettra de résoudre des instances plus grandes. Cette fonction sera utilisée pour évaluer la qualité des algorithmes quantiques.

▶ **Correction**

Modèle mathématique :

$$\max_{x \in \{0,1\}^n} \sum_{i=1}^{n} u_i x_i$$

$$s.t. \sum_{i=1}^{n} v_i x_i = V$$

2 Résolution avec QAOA

QAOA nécessite en entrée un problème sous forme QUBO (Quadratic Binary Optimization), c'est-à-dire minimiser une fonction quadratique à variables binaires sans contrainte. L'évacuation des contraintes du problème se fait par leur intégration dans la fonction objectif de la façon suivante : étant donné une contrainte $\mathcal{C}$ de la forme $g(x) = 0$, trouver un fonction de pénalité $p_{\mathcal{C}}$ (qui peut être non linéaire) telle que

$$\begin{cases} p_{\mathcal{C}}(x) = 0, & \text{si } x \text{ satisfait la contrainte } \mathcal{C}, \text{ i.e. } g(x) = 0 \\ p_{\mathcal{C}}(x) > 0, & \text{si } x \text{ ne satisfait pas la contrainte } \mathcal{C}, \text{ i.e. } g(x) \neq 0 \end{cases}$$

Ensuite, additionner les termes de pénalités de chaque contrainte du problème initial à la fonction objectif initiale, chacun multiplié par un réel $\lambda_{\mathcal{C}}$ appelé coefficient de pénalité.

4. Reformuler le problème de sac à dos sous forme QUBO. Vous expliciterez le choix des termes de pénalités pour chaque type de contrainte, ainsi que le signe des coefficients de pénalités.

5. Effectuer un changement de variable pour remplacer les variables binaires par des variables $\{-1, 1\}$.

6. Écrire l'Hamiltonien H_C associé, montrer sur un exemple non trivial de votre choix qu'une solution est bien associée à un vecteur propre de valeur propre λ égale à l'opposé de la somme des utilités des objets qu'on place dans le sac.

7. Décrire le circuit quantique de la matrice unitaire $\exp i\gamma H_C$.

   On va maintenant pouvoir s'attaquer au code.

   Conseil pour coder QAOA avec Qiskit:

   - Si vous avez bien compris l'algorithme, vous devriez pouvoir coder vous même l'algorithme. Mais ce serait un peu laborieux. Qiskit vous propose une boîte noire toute faite pour cela. Cette boîte noire nécessaire malgré tout l'hamiltonien.
   - Vous pouvez décrire un hamiltonien très facilement avec Qiskit. On peut encoder une porte $A_1 \otimes A_2 \otimes \cdots \otimes A_n$ où $A_i \in \{Z, I\}$ avec la chaîne de caractères $A_n A_{n-1} \cdots A_2 A_1$ (où A_i est remplacé par le caractère `Z` ou `I` selon sa valeur). Attention à l'ordre des portes!

     Par exemple la porte $Z \otimes Z \otimes I$ pourra être encodée avec la chaîne `IZZ`.

     Ensuite, un hamiltonien $H_C = \alpha \cdot A_1 \otimes A_2 \otimes \cdots \otimes A_n + \beta B_1 \otimes B_2 \otimes \cdots \otimes B_n + \cdots + \gamma C_1 \otimes C_2 \otimes \cdots \otimes C_n$ pourra être construit avec la fonction suivante:
     `SparsePauliOp.from_list([("`$A_n A_{n-1} \cdots A_1$`", `α`), ("`$B_n B_{n-1} \cdots B_1$`", `β`), ..., ("`$C_n C_{n-1} \cdots C_1$`", `γ`)])`

     Par exemple $H_C = Z \otimes Z \otimes I + 2I \otimes Z \otimes I$ pourra être créé avec
     `SparsePauliOp.from_list([("IZZ", 1), ("IZI", 2)])`
   - Une fois l'hamiltonien connu, vous pouvez construire le circuit de QAOA avec l'opérateur `QAOAAnsatz` qui prend en entrée l'hamiltonien et un paramètres `reps` égal au nombre de répétition des alternances des exponentielles des portes H_C et H_B.
   - Rappelez vous que QAOA construit un circuit paramétré, il faut, pour le faire fonctionner donner une valeur à ses paramètres. Il faut ensuite trouver le bon set de paramètres pour que ce circuit vous donne la meilleure solution. Vous devez donc optimiser les paramètres du circuit avec un algorithme classique. Vous pouvez utiliser un optimiseur déjà conçu pour ça.

     Commencez par récupérer le nombre de paramètres du circuit:

     `q.num_parameters`

     Fixez ensuite ces paramèters à des valeurs aléatoires, par exemple entre 0 et 2π.

     Il nous faut une fonction capable d'évaluer un set de paramètres `params`. On utilisera la fonction suivante:

```
from qiskit.primitives import StatevectorEstimator

def f(params):
    """
    On suppose qu'on a deux variables globales.
    q est le circuit paramétré renvoyé par QAOAAnsatz
    hamiltonian est l'hamiltonien généré avec la fonction SparsePauliOp.from_list
    """

    pub = [q, [hamiltonian], [params]]
    estimator = StatevectorEstimator()
    result = estimator.run(pubs=[pub]).result()
    cost = result[0].data.evs[0]

    return cost
```

Ce circuit évalue la qualité des paramètres `params` au travers de l'hamiltonien. Pour simplifier grossièrement, la fonction `estimator.run` effectue les opérations suivantes:

- Instancier les paramètres du circuit `q` avec les valeurs données dans la liste `params`
- Exécuter le circuit `q` de nombreuses fois pour produire une distribution de qbits
- Tester chaque vecteur de cette distribution sur l'hamiltonien pour déterminer la valeur propre moyenne sur cette distribution. Plus la valeur propre moyenne est petite, plus le circuit a de grandes chances de sortir une solution réalisable de bonne qualité.

Plus d'informaton dans le premier TP sur qiskit.

On peut ensuite utiliser ensuite la fonction de `scipy` nommée `minimize` pour trouver un bon set de paramètres.

`best_params = minimize(f, init_params, args=(), method="COBYLA").x`

Enfin, on peut rappeler `f` avec ces paramètres pour avoir la valeur objective finale ou mesurer manuellement la distribution finale et extraire la meilleure solution. Pour instancier manuellement les paramètres, il faut utiliser:

`q.assign_parameters`

8. Ecrire un code, commenté, résolvant le problème de sac à dos avec l'algorithme QAOA.

3 Résolution avec l'algorithme de Grover

Afin de résoudre le problème de sac à dos avec l'algorithme de Grover, nous allons commencer par résoudre le problème de décision associé.

3.1 Résolution du problème de décision associé

L'oracle de l'algorithme de Grover associé au problème de *décision* de sac à dos se décompose en deux parties. La première partie marque uniquement les solutions réalisables, tandis que la seconde partie sélectionne les solutions dont la valeur est supérieure à un entier fixé k.

9. Écrire le problème de décision associé au problème de sac à dos.

10. Proposer schématiquement la construction de l'oracle du problème de décision, en précisant le rôle de chaque partie de l'oracle.

11. Proposer un circuit pour l'instance suivante et la valeur $k = 3$. Vous pouvez supposer que vous disposer d'une porte S qui calcule la somme de deux entiers, d'une porte Eq qui vérifie si deux entiers sont égaux et d'une porte In qui vérifie si un entier est supérieure ou égal à un autre entier (les entrées et sorties de ces portes peuvent être décrites librement dans votre réponse).

    $V = 2, v_1 = 2, v_2 = 1, v_3 = 1, u_1 = 1, u_2 = 2, u_3 = 1.$

12. Pour ne pas exploser le nombre de qbits avec les qbits de travail dans qiskit, on va tricher un peu. Soit une liste L de p nombres entre 0 et $2^n - 1$, expliquer comment construire un circuit de profondeur $O(|L|)$ qui, connaissant un qbit de base $|\underline{x}\rangle$ renvoie $-|\underline{x}\rangle$ si $x \in L$ et renvoie $|\underline{x}\rangle$ sinon.

13. Implanter ensuite une fonction qui commence par construire L la liste de toutes les solutions réalisables dont le poids est supérieur à k. Utiliser ensuite cette fonction pour construire un oracle.

14. Expliquer pourquoi concevoir ce circuit est en contradiction avec l'utilisation de l'algorithme de Grover.

15. Codez (malgré la contradiction) avec qiskit un circuit qui implante l'algorithme de Grover pour résoudre le problème de décision associé au problème de sac à dos.

3.2 Résolution du problème d'optimisation

Nous revenons au problème initial, qui est le problème d'optimisation du sac à dos.

16. Proposer de façon schématique le fonctionnement de l'algorithme d'optimisation.

17. Implémenter cet algorithme avec Qiskit.

4 Evaluation

Proposer une évaluation de la qualité des 2 algorithmes, QAOA et Grover. Cet algorithme évaluera:

- le temps de calcul
- la qualité des solutions renvoyées, utilisez pour cela l'algorithme exact codé au début du projet.
- la complexité des algorithmes (nombre de porte, profondeur des circuits, nombre de qbits, nombre de répétition des algorithmes quantiques).