LABORATOIRE D'INFORMATIQUE FONDAMENTALE D'ORLEANS

Université d'ORLÉANS

M2 informatique + GPEX Minerve

TD programmation quantique : utilisation du quantique (algorithme de Grover) comme boîte noire

Nicolas OLLINGER et Ioan TODINCA

1 Amplification des algorithmes probabilistes

Soit P un problème de décision. Soit $\mathcal{A}$ un algorithme probabiliste avec les propriétés suivantes :
- Si l'algorithme répond true pour l'entrée I, alors la réponse est correcte.
- Si l'algorithme répond false pour l'entrée I, alors la réponse est correcte avec probabilité au moins $\varepsilon > 0$, où ε est une constante indépendante de I.
- La complexité de $\mathcal{A}$ est $T(n)$ pour une certaine fonction T, n étant la taille de l'entrée.

1. Considérons l'algorithme de Grover pour une fonction quelconque, qui fait *un seul appel* au circuit de Grover avec un t choisi uniformément au hasard parmi $\{1, \ldots, \pi\sqrt{N}/4\}$ où $N = 2^n$. Montrer que cet algorithme, **modulo un petit patch**, peut être interprété avec les notations ci-dessus, en précisant les valeurs de ε et de la fonction T.
2. Supposons pour cette question que $\varepsilon = 1/2$. Donner un algorithme $\mathcal{A}_{bon}$ qui obtient la réponse correcte avec probabilité $\geq 1 - 1/n$, puis un algorithme $\mathcal{A}_{super}$ assurant la réponse correcte avec probabilité $\geq 1 - 1/2^n$. Préciser leurs complexités respectives, T_{bon} et T_{super}.
3. Même question que précédemment, mais pour une valeur $\varepsilon > 0$ quelconque. Observer l'ordre de grandeur du nombre d'appels à l'algorithme $\mathcal{A}$ et des fonctions T_{bon} et T_{super}.
4. Repréciser l'algorithme de Grover vu en TP, pour une fonction quelconque, en assurant qu'il trouve la bonne réponse avec probabilité au moins $1 - 1/2^n$. Rappeler sa complexité en temps.

2 Calculer $\arg\min$ ou $\arg\max$ avec la même complexité que Grover

Soit $f : \{0, \ldots N-1\} \to \{0, \ldots N-1\}$ une fonction entière, pour les entiers codés sur n bits (donc $N \leq 2^n$). Nous disposons d'une implémentation classique de f, et d'un compilateur efficace transformant n'importe quelle fonction classique f en circuit quantique U_f. Le premier objectif est de trouver $x \in \{0, \ldots N-1\}$ tel que $f(x)$ soit minimum, avec un algorithme mixte classique/quantique en temps $O(\sqrt{N})$, avec la garantie que le minimum est trouvé avec probabilité au moins $1/2$.

Algorithm 1 Algorithme calculant $\arg\min f$ pour une fonction $f : \{0, \ldots N-1\} \to \{0, \ldots N-1\}$

```
choisir x uniformément au hasard dans {0, ... N-1}
while la complexité totale des appels à Grover est ≤ c√N pour une certaine constante c do
    chercher, avec Grover, un y ∈ {0, ..., N-1} tel que f(y) < f(x)
    x ← y
end while
return x
```

L'algorithme 1 est dû à C. Dürr et P. Høyer, dans une version que vous trouverez sur ArXiV.

1. Comment implémenter l'instruction de la ligne 3 ? (Ce n'est pas exactement la version de Grover vue en cours et TP dont en a besoin ici, mais une variante de Boyer, Brassard, Høyer et Tapp, trouvant une solution en $O(\sqrt{N/b})$ étapes en moyenne, où b est le nombre de solutions.)
2. C. Dürr et P. Høyer ont précisé la constante c (=23 pour N suffisamment grand) et ont prouvé que l'algorithme 1 trouve le minimum avec probabilité au moins $\frac{1}{2}$. Comment booster l'algorithme afin que la probabilité de trouver la bonne réponse soit au moins $1 - \frac{1}{2^n}$?

3 Connexité d'un graphe en temps sous-linéaire

Considérons un graphe $G = (V, E)$ non orienté avec N sommets et M arêtes. L'objectif est donner un algorithme vérifiant si le graphe est connexe (ou calculant un arbre recouvrant de poids minimum) en temps $\tilde{O}(\sqrt{NM})$, en utilisant le calcul quantique. La notation $\tilde{O}$ ignore les facteurs polynomiaux en $\log N$.

1. Pourquoi une telle complexité est-elle surprenante ?

Dürr, Heiligman, Høyer et Mhalla généralisent la recherche du minimum de f à la recherche de $\ll d$ plus petits éléments de f de type différent $\gg$ en temps $O(\sqrt{dN})$. Le problème consiste à considérer une fonction f comme dans l'exercice précédent, mais aussi un type $Type[x] \in \{1, \ldots, d\}$ pour chaque $x \in \{0, \ldots, N-1\}$ et de retourner simultanément le minimum pour chaque type. Ils proposent l'algorithme 2 calculant un arbre recouvrant de poids minimum d'un graphe, dont les arêtes ont des poids différents, codés sur $n = \log N$ bits. C'est une variante quantique de l'algorithme de Borůvka.

Algorithm 2 Algorithme de Borůvka quantique. Calcule un arbre recouvrant de poids minimum

1: $T_1, \ldots, T_k$ est une forêt couvrante. Initialement $k = N$ et chaque T_i contient le sommet i.
2: $\ell \leftarrow 0$
3: **while** $k > 1$ **do**
4: $\quad \ell \leftarrow \ell + 1$
5: $\quad$ Chercher les arêtes $e_1, \ldots, e_k$ telles que e_j est l'arête de poids minimum reliant
6: $\quad\quad T_j$ à l'extérieur. Interrompre la recherche lorsque la complexité totale atteint
7: $\quad\quad (\ell + 2)c\sqrt{kM}$ pour une constante c bien choisie.
8: $\quad$ Fusionner les arbres reliés par les arêtes e_j et mettre k à jour.
9: **end while**

2. Observer qu'à chaque itération, à la ligne 5 on a $k \leq N/2^{\ell-1}$ et que la complexité en temps de l'instruction est $O((l+2)c\sqrt{NM/2^{\ell-1}})$.
3. Montrer que le nombre d'itérations de la boucle est $O(\log N)$. Montrer (ou faire confiance à Dürr *et al.*) que la complexité totale de l'algorithme est $\tilde{O}(\sqrt{NM})$.
4. En admettant que la probabilité d'erreur à l'itération ℓ est au plus $\frac{1}{2^{\ell+2}}$, montrer que la probabilité d'erreur de l'algorithme est au plus $\frac{1}{4}$.
5. Adapter l'algorithme pour vérifier si le graphe G est connexe. On pourra utiliser comme poids de l'arête xy la quantité $xN + y$, pour chaque $x, y \in \{0, \ldots, N-1\}$, avec la convention $x < y$.