Bornes inférieures en calcul quantique : Méthode par adversaire vs. méthode des polynômes

Pierre Philipps

Rapport de stage de DEA, effectué au LRI sous la direction de Frédéric Magniez

Introduction

Le calcul quantique est un modèle de calcul prenant en compte des phénomènes physiques particuliers dus à la mécanique quantique, permettant d'obtenir des algorithmes plus efficaces pour différents problèmes ; en particulier, l'algorithme de Grover[8], réalisant la recherche d'un élément dans une liste non triée, permet d'obtenir une accélération quadratique par rapport à n'importe quel équivalent classique. L'algorithme de factorisation de Shor[17] en temps polynomial est également prometteur pour d'éventuelles applications en cryptanalyse, dès que la réalisation pratique d'un ordinateur quantique aura dépassé le stade expérimental ; en effet, le meilleur algorithme classique connu résolvant le même problème travaille en temps exponentiel. Cependant, malgré l'intérêt énorme de ces résultats, il est naturel de s'interroger sur leur optimalité ; c'est la raison pour laquelle l'étude de bornes inférieures de complexité dans ce modèle quantique est très importante. C'est pourquoi nous avons étudié, lors de ce stage de DEA, différentes techniques d'obtention de telles bornes inférieures, et leur efficacité respective, à travers un certain nombre de problèmes liés à l'algorithme de Grover.

Après avoir présenté le modèle de calcul quantique, nous exposerons les deux principales et plus puissantes méthodes utilisées pour établir des bornes inférieures de complexité en calcul quantique : la méthode par adversaire quantique[2, 11], et la méthode des polynômes[4, 15, 12, 1]. Ceci nous amènera à modifier le contexte, rendant cette deuxième méthode sous-optimale par rapport à la première, ce qui sera montré à travers des exemples de problèmes présentés en dernière partie. Ceci montre donc la supériorité de la méthode par adversaire sur la méthode des polynômes dans ce nouveau modèle, dit modèle de test.

1 Modèle de calcul quantique

Dans ce rapport, nous nous intéressons uniquement à l'aspect théorique de ce modèle. Les problèmes d'ordre physique ne sont pas étudiés.

1.1 Définitions et notations

1.1.1 Bit quantique

Un bit classique peut prendre deux valeurs différentes : 0 et 1. Intuitivement, le bit quantique, ou qubit, peut être dans ces deux états simultanément, formant une superposition quantique. En utilisant la notation bra-c-ket de Dirac[1], un qubit ϕ s'exprime alors sous la forme :

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Dans cette expression, α et β sont des nombres complexes dont la somme des normes au carré est égale à 1, et qui représentent *l'amplitude* des états $|0\rangle$ et $|1\rangle$ présente dans $|\phi\rangle$. Mathématiquement, $|\phi\rangle$ est donc un vecteur normé d'un $\mathbb{C}$-espace de Hilbert de dimension 2, dont $|0\rangle$ et $|1\rangle$ forment une base orthonormale ; cet espace sera noté H_2.

1.1.2 Mesure d'un qubit

On peut ainsi penser pouvoir stocker plus d'information dans un qubit que dans un bit classique[2] ; cependant, cette information est également beaucoup plus difficile d'accès. En effet, le seul moyen d'accéder à la valeur d'un qubit est d'effectuer une *mesure* ; et les seuls résultats possibles d'une telle mesure sont les états classiques dont le qubit est une superposition, c'est-à-dire ici $|0\rangle$ ou $|1\rangle$. La probabilité d'obtenir l'un ou l'autre de ces deux résultats dépend directement et uniquement de l'amplitude associée à chacun de ses états : si $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, on obtiendra $|0\rangle$ avec probabilité $|a|^2$ et $|1\rangle$ avec probabilité $|b|^2$ (comme $|\phi\rangle$ est normé, la somme de ces deux probabilités est bien égale à 1).

Enfin, une dernière chose fondamentale, et qui rend une telle opération problématique : *l'observation perturbe la mesure*. Plus précisément, après la mesure, $|\phi\rangle$ se retrouve dans l'état dans lequel on l'a mesuré ; c'est-à-dire $|0\rangle$ ou $|1\rangle$, suivant ledit résultat. Ainsi, contrairement à un bit probabiliste classique, où la mesure n'influe pas sur l'état du bit en question, il est impossible de retrouver de manière probabiliste les normes des amplitudes α et β en effectuant plusieurs mesures consécutives, puisque ces amplitudes disparaissent ; à partir de la deuxième mesure, on trouvera toujours le premier état mesuré ($|0\rangle$ ou $|1\rangle$) avec probabilité 1.

Notons également qu'il est en fait possible d'effectuer une mesure dans n'importe quelle base orthonormale, dans laquelle il faut alors décomposer $|\phi\rangle$ pour trouver les probabilités de mesure des différents éléments de cette nouvelle base.

1.1.3 Généralisation à un espace quelconque

Au lieu de se limiter à un qubit, il est possible de considérer n'importe quel système physique pouvant classiquement être dans N états différents notés $|1\rangle, \ldots, |N\rangle$. Dans ce cas, un état quantique

[1] Dans cette notation, un vecteur colonne v est dans un "ket" $|v\rangle$, tandis qu'un vecteur ligne w se retrouve dans un "bra" $\langle w|$ - plus précisément $\langle w|$ représente le transconjugué de $|w\rangle$; le produit scalaire des deux vecteurs est alors noté $\langle w|v\rangle$ tandis qu'une application projetant w en v est naturellement notée $|v\rangle\langle w|$.

[2] Ce n'est pas le cas ; n qubits sont nécessaires pour stocker autant d'information accessible que dans n bits classiques[9].

de ce système est un vecteur normé d'un $\mathbb{C}$-espace de Hilbert H dont les vecteurs $|1\rangle, \dots, |N\rangle$ forment une base orthonormale. Une mesure d'un état quantique dans cette base se fait de manière similaire au cas d'un qubit.

Mais il est également possible de définir une mesure de manière beaucoup plus générale. Ainsi, si l'on considère une décomposition de l'espace H en sous-espaces orthogonaux $V_1, \dots, V_M$, et si l'on note P_j la projection orthogonale sur V_j, alors tout vecteur (ou état quantique) $|\phi\rangle$ de H s'écrit :

$$|\phi\rangle = \sum_{j=1}^{M} P_j|\phi\rangle$$

Cette décomposition définit une mesure, dont le résultat sera $P_j|\phi\rangle$ avec probabilité $\|P_j|\phi\rangle\|^2$. Comme avant, l'état mesuré devient le nouvel état quantique du système. Cette notion de mesure est clairement une généralisation de la précédente, puisque toute base orthonormale définit une décomposition de H en autant de sous-espaces orthogonaux.

1.1.4 n qubits

Le passage de 1 qubit à n qubits se fait par l'intermédiaire d'un *produit tensoriel*[3], noté $\otimes$. Un système physique formé de n qubits est ainsi représenté par un vecteur $|\phi\rangle$ évoluant dans le produit tensoriel de n $\mathbb{C}$-espaces de Hilbert H_2. Ce produit est noté H_{2^n} et possède une base orthonormale formée des produits tensoriels des vecteurs de la base de départ : $|e_1\rangle \otimes |e_2\rangle \otimes \dots \otimes |e_n\rangle$, où chacun des e_i peut prendre la valeur 0 ou 1. Pour alléger les notations, on omettra la plupart du temps le symbole $\otimes$, et on pourra ainsi utiliser l'une des trois notations suivantes : $|e_1\rangle|e_2\rangle \dots |e_n\rangle$, $|e_1, e_2, \dots, e_n\rangle$, ou encore $|e_1 e_2 \dots e_n\rangle$; remarquons que, comme chacun des e_i représente en fait un bit d'information, il est tout à fait possible de considérer les notations ci-dessus comme l'écriture en binaire d'un nombre, et l'on pourra donc à l'occasion utiliser l'écriture décimale de ce nombre, à condition bien sûr que cela ne crée pas d'ambiguïté. Par exemple, la base principale d'un système à 2 qubits pourra être notée indifféremment $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$ ou $(|0\rangle, |1\rangle, |2\rangle, |3\rangle)$.

Ainsi, notre système quantique à n qubits $|\phi\rangle$ est une superposition de 2^n états classiques, et possède donc 2^n coordonnées complexes. Les amplitudes de cette superposition déterminent comme précédemment les probabilités d'obtenir un des 2^n résultats possibles, si l'on mesure l'ensemble de ces n qubits. Mais on peut également mesurer un seul de ces qubits, en considérant les espaces de projection $|0\rangle \otimes H_{2^{n-1}}$ et $|1\rangle \otimes H_{2^{n-1}}$, ou même n'importe quel sous-ensemble de qubits. Remarquons qu'il est équivalent, d'un point de vue calculatoire, de mesurer un ensemble de qubits d'un seul coup, ou qubit par qubit (dans n'importe quel ordre)[4].

Pour illustrer ces définitions, considérons l'exemple du paradoxe EPR (Einstein, Podolsky et Rosen), où l'on considère le système à 2 qubits suivant :

$$|\phi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

[3] Le produit tensoriel de deux espaces de Hilbert H et H', de dimensions respectives n et m, et ayant respectivement comme bases orthonormales $(|e_i\rangle)_{1 \le i \le n}$ et $(|e'_j\rangle)_{1 \le j \le m}$, est l'espace $H \otimes H'$ de dimension nm, dont une base orthonormale est l'ensemble des vecteurs $(|e_i\rangle \otimes |e'_j\rangle)_{1 \le i \le n, 1 \le j \le m}$. Malgré les apparences, $H \otimes H'$ est unique à isomorphisme près. Le produit tensoriel est associatif, non commutatif, distributif par rapport à l'addition de vecteurs, et homogène par multiplication par un scalaire. Enfin, le produit scalaire d'un produit tensoriel est égal au produit des produits scalaires : $(\langle w_1| \otimes \langle w_2|)(|v_2\rangle \otimes |v_1\rangle) = \langle w_1|v_1\rangle\langle w_2|v_2\rangle$.

[4] On peut également montrer que tout algorithme quantique est équivalent à un autre algorithme où les mesures ne sont effectuées qu'à la fin de ce dernier.

Dans cet exemple très simple, si l'on mesure le premier qubit, pour connaître les différents résultats possibles et les probabilités associées, il faut décomposer $|\phi\rangle$ suivant ce qubit :

$$|\phi\rangle = |0\rangle \otimes \left(\frac{|0\rangle}{\sqrt{2}}\right) + |1\rangle \otimes \left(\frac{|1\rangle}{\sqrt{2}}\right)$$

Ainsi, on obtiendra le résultat $|0\rangle$ avec probabilité $\left\|\frac{|0\rangle}{\sqrt{2}}\right\|^2 = (1/\sqrt{2})^2 = 1/2$, auquel cas $|\phi\rangle$ se retrouvera dans l'état $|00\rangle$; de même pour l'autre résultat possible $|1\rangle$. Remarquons qu'après avoir effectué une telle mesure, l'ensemble du système se retrouve dans un état classique où les deux qubits ont la même valeur. Ainsi, la mesure de l'un des 2 qubits de cette paire force l'autre à partager instantanément le même état résultat, alors même que le-dit résultat est aléatoire, et ceci sans communication entre les 2 qubits ; on a affaire à ce qu'on appelle un état *enchevêtré*, c'est-à-dire ne pouvant pas se décomposer sous la forme d'un produit tensoriel. Cette notion d'enchevêtrement fait qu'il est impossible de considérer un système composé de plusieurs qubits comme la juxtaposition de ces qubits ; c'est pourquoi on parle parfois d'un n-qubit plutôt que de n qubits pour illustrer cet état de fait.

En apparence, en tout cas dans cet exemple, cette notion d'enchevêtrement semble équivalente à la présence d'une variable aléatoire cachée ; cependant, une telle paire de qubits permet de réaliser des choses impossibles en classique. Par exemple, si Alice veut envoyer l'information contenue dans deux bits classiques à Bob, le fait de partager une variable aléatoire ne peut pas les aider, et Alice doit nécessairement envoyer 2 bits d'information à Bob ; mais dans le modèle quantique, si Alice et Bob partagent une paire EPR, l'envoi d'un unique qubit suffit à coder 2 bits classiques[5].

1.1.5 Evolution unitaire

En dehors de l'opération de mesure d'un ensemble de qubits, les seules opérations autorisées (pour des raisons physiques) sont des transformations unitaires, c'est-à-dire réversibles, linéaires et conservant la norme. Notons que pour des raisons de linéarité, il suffit pour définir entièrement une telle opération de décrire son action sur n'importe quelle base ; par exemple, si U est une transformation quantique agissant sur n qubits, il suffit pour la définir de donner $U|i\rangle$ pour i variant de 0 à $2^n - 1$.

C'est grâce à ce type d'opérations que l'on réalise en pratique une mesure dans n'importe quelle base ; en effet, si l'on considère deux bases orthonormales $\mathcal{B} = (|\psi_1\rangle, \ldots, |\psi_N\rangle)$ et $\mathcal{B}' = (|\psi'_1\rangle, \ldots, |\psi'_N\rangle)$, $M_{\mathcal{B}}$ et $M_{\mathcal{B}'}$ les opérateurs de mesure correspondant à chacune de ces bases, et $V_{\mathcal{B}\to\mathcal{B}'}$ la transformation unitaire définie par $V_{\mathcal{B}\to\mathcal{B}'}|\psi_i\rangle = |\psi'_i\rangle$ pour tout i, alors on vérifie facilement que l'on a la relation suivante :

$$M_{\mathcal{B}'} = V_{\mathcal{B}\to\mathcal{B}'} M_{\mathcal{B}} (V_{\mathcal{B}\to\mathcal{B}'})^{-1}$$

En particulier, si l'on considère $\mathcal{B} = (|1\rangle, \ldots, |N\rangle)$ la base canonique, cela nous permet d'implémenter la mesure dans n'importe quelle base $\mathcal{B}'$. Remarquons ici que la difficulté d'une telle mesure est alors contenue dans la difficulté de réaliser la porte $V_{\mathcal{B}\to\mathcal{B}'}$ et son inverse.

1.2 Circuits quantiques

Il y a différents modèles de calcul (en général équivalents) prenant en compte les spécificités quantiques ; le plus facile à utiliser dans la plupart des cas est le modèle de circuits quantiques, que nous allons donc présenter.

[5] Comme Alice et Bob partagent une paire EPR, cela ne contredit pas le résultat mentionné précédemment comme quoi on ne peut pas coder n bits classiques dans $n - 1$ qubits.

1.2.1 Porte et circuit quantique

Une porte quantique est une transformation unitaire agissant sur un nombre constant de qubits (en général 1, 2 ou 3). Par exemple, la porte d'Hadamard, notée H, agit sur 1 qubit de la manière suivante :

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$
$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

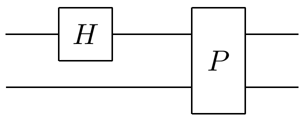
Ces portes peuvent être composées par produit tensoriel (car le produit tensoriel de transformations unitaires reste unitaire) pour les effectuer en parallèle sur des registres différents. Dans le cas de la porte d'Hadamard, si on l'applique en parallèle sur n qubits, on obtient le résultat suivant :

$$H^{\otimes n}|i\rangle = \frac{1}{2^n} \sum_{j \in \{0,1\}^n} (-1)^{i.j}|j\rangle$$

(où $i.j$ représente le produit scalaire modulo 2 des chaînes de bits i et j dans $(\mathbb{Z}_2)^n$.)

En particulier, sur l'entrée[6] $|0\rangle^{\otimes n}$, $H^{\otimes n}$ renvoie une superposition uniforme de tous les états possibles ; c'est pourquoi la porte d'Hadamard joue souvent un rôle important dans la plupart des algorithmes quantiques.

Les portes quantiques peuvent également être composées par produit matriciel pour les effectuer en série :



Ici, P représente une porte quelconque agissant sur 2 qubits ; le circuit ci-dessus réalise alors la transformation suivante : $P(H \otimes I)$, où I est l'identité sur 1 qubit.

Un circuit quantique se définit de manière similaire à son équivalent classique à partir de ces portes élémentaires. Il s'agit donc d'un graphe orienté fini acyclique, formé de portes, d'un certain nombre d'entrées et du même nombre de sorties (puisque toutes les portes utilisées sont réversibles).

1.2.2 Universalité

Une question que l'on peut naturellement se poser à présent est celle de savoir s'il est possible de calculer par un circuit utilisant un ensemble donné de portes (agissant sur un nombre borné de qubits) une transformation unitaire quelconque donnée. On s'intéresse en particulier à des ensembles de portes permettant de calculer n'importe quelle transformation unitaire ; cela pourrait donner naissance à une notion d'universalité, mais cette dernière serait trop restreinte. Par exemple, on aimerait pouvoir utiliser des *bits de travail* ; on dira ainsi qu'un circuit réalise la transformation unitaire à n qubits U si C réalise en fait la transformation unitaire à m qubits V $(m > n)$ avec la propriété suivante :

$$\forall\ |\phi\rangle \in H_{2^n},\ V(|\phi\rangle|0\rangle^{\otimes(m-n)}) = (U|\phi\rangle)|0\rangle^{\otimes(m-n)}$$

[6] $|0\rangle^{\otimes n}$ est le produit tensoriel de n qubits dans l'état $|0\rangle$; on omettra généralement le symbole de produit tensoriel dans l'exposant ($|0\rangle^n$), car cela ne crée pas d'ambiguïté. Par contre, si l'on considère une porte quantique U, U^n (succession de n portes U en série) est très différent de $U^{\otimes n}$ (mise en parallèle de n portes U) ; par conséquent, il faudra toujours faire la distinction entre les deux.

Les $m - n$ qubits dans l'état $|0\rangle$ représentent ainsi un espace de travail utilisé pour calculer la transformation unitaire U ; comme indiqué par l'équation ci-dessus, ces qubits supplémentaires doivent impérativement être revenu dans leur état initial après l'action de V, ceci afin d'éviter des problèmes d'interférence (autrement dit d'enchevêtrement) avec les autres qubits.

Par ailleurs, comme il existe une infinité non dénombrable de portes différentes, il est impossible de toutes les simuler à partir d'un ensemble fini. Cependant, si on ne se limite pas à des ensembles finis, tout est possible :

Théorème 1.1 ([3]) *L'ensemble constitué de la porte $C - NOT$[7] et de toutes les portes agissant sur* 1 *qu-bit permet de simuler exactement n'importe quelle transformation unitaire.*

Mais dans l'optique d'une réalisation pratique, il est préférable de se limiter à des ensembles finis. C'est pourquoi on se contente en général de pouvoir approcher avec n'importe quelle précision les transformations unitaires à simuler, plutôt que de les réaliser de manière exacte ; ainsi, on dit qu'un circuit calculant la transformation unitaire V réalise la porte U à ϵ près (et avec $m - n$ qubits de travail) si et seulement si :

$$\forall\ |\phi\rangle \in H_{2^n},\ \left\| V(|\phi\rangle|0\rangle^{\otimes(m-n)}) - (U|\phi\rangle)|0\rangle^{\otimes(m-n)} \right\| \leq \epsilon$$

Tout ceci nous amène à la définition d'universalité suivante :

Définition 1.2 (Ensemble de portes universel) *Un ensemble de portes est dit* universel *si et seulement si toute transformation unitaire peut être approchée avec n'importe quelle précision par un circuit construit à partir de cet ensemble de portes et utilisant un nombre quelconque de bits de travail.*

Quelques exemples d'ensembles de portes universels à présent :

Théorème 1.3 *[6] L'ensemble de portes constitué de $C - NOT$, de la porte d'Hadamard définie précédemment et de la porte de changement de phase $R_{\pi/4}$[8] est universel.*

Théorème 1.4 *[16] L'ensemble de portes formé de la porte de Toffoli[9] et de la porte d'Hadamard H est universel.*

Ce dernier théorème est intéressant, car la porte de Toffoli est connue pour être universelle en calcul classique réversible.

1.2.3 Calcul d'une fonction

Jusqu'à présent, nous nous sommes contentés de voir comment simuler des transformations unitaires avec des portes quantiques ; cela ne nous donne pas de modèle de calcul à proprement parler, puisque l'on ne sait pas ce que l'on appelle calculer une fonction. On se donne donc une fonction $f : \{0, \ldots, 2^n - 1\} \rightarrow \{0, \ldots, 2^m - 1\}$ (c'est-à-dire une fonction avec n bits en entrée et m bits en sortie), et un circuit C définissant une transformation unitaire U agissant sur N qubits.

[7] Cette porte agit sur 2 qubits en inversant le seconde qubit si et seulement le premier est dans l'état $|1\rangle$; de manière équivalente, $C - NOT$ agit de la manière suivante : $|x\rangle|y\rangle \rightarrow |x\rangle|x \oplus y\rangle$, où $\oplus$ représente le OU-exclusif.

[8] Le changement de phase R_θ agit sur 1 qubit de la manière suivante : $R_\theta|0\rangle = |0\rangle$ et $R_\theta|1\rangle = e^{\imath\theta}|1\rangle$.

[9] La porte de Toffoli, notée T, agit sur 3 qubits de la manière suivante : $T|a, b, c\rangle = |a, b, c \oplus (a \wedge b)\rangle$.

Définition 1.5 *On dit que la transformation unitaire U calcule f si et seulement si pour toute entrée $x \in \{0, \ldots, 2^n - 1\}$, on a :*

$$\sum_{z,z'} \left| \langle z', F(x), z | U | x, 0^m, 0^{N-n-m} \rangle \right| \geq 1 - \epsilon$$

où ϵ est un réel positif strictement plus petit que $1/2$.

Cette expression signifie exactement que la mesure d'un certain ensemble de qubits à la fin de l'application du circuit sur l'entrée $|x, 0^m, 0^{N-n-m}\rangle$ renvoie $F(x)$ avec probabilité plus grande que $1 - \epsilon$. Cette définition est similaire à celle du modèle de calcul probabiliste.

1.2.4 Simulation de calcul classique en quantique

Si on se donne maintenant une fonction de $\{0, \ldots, 2^n - 1\}$ dans $\{0, \ldots, 2^m - 1\}$, on se demande naturellement comment réaliser en pratique un circuit calculant f. La solution la plus simple est de faire appel à un résultat classique[5] disant que si l'on sait calculer f par un circuit classique utilisant s portes booléennes, alors on peut calculer $f_{\text{rev}} : (x, y) \rightarrow (x, y \oplus f(x))$ par un circuit réversible utilisant $O(s)$ portes de Toffoli[10].

Pour passer ensuite au modèle de calcul quantique, on remarque que f_{rev} réalise une permutation des éléments de la base canonique, et peut donc être étendue de manière linéaire en une opération unitaire à toute superposition de ces vecteurs de base. On note U_f la transformation quantique ainsi obtenue ; U_f agit sur $n + m$ qubits de la manière suivante :

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

Il est facile de voir que U_f, ou tout circuit approchant U_f, calcule bien f. Ainsi, il est possible d'utiliser le résultat précédemment cité pour réaliser en quantique avec une complexité équivalente n'importe quelle fonction classique donnée par un circuit booléen. Cependant, il faut faire attention, car la fonction réversible f_{rev} produite par le théorème de Bennett utilise en fait des bits supplémentaires qui sont ensuite jetés à la poubelle. Cela ne gène pas dans la définition de calcul que l'on a vu ci-dessus ; cependant, si l'on veut ensuite utiliser cette fonction f dans un calcul, les qubits que l'on élimine ainsi pourraient par la suite continuer à interférer avec le résultat du calcul. Ainsi, on sait en fait réaliser la porte suivante :

$$U'_f |x\rangle |y\rangle |0\rangle^{\otimes k} = |x\rangle |y \oplus f(x)\rangle |g(x, y)\rangle$$

où $g(x, y)$ représente k qubits de déchet à éliminer. U'_f calcule bien f, mais avec déchets, ce que l'on aimerait éviter. Heureusement, il est possible d'obtenir effectivement U_f à partir de U'_f en utilisant des qubits de travail supplémentaire de la manière suivante, où l'on va copier[11] le résultat à l'aide de portes $C - NOT$ du deuxième au quatrième registre, puis appliquer $U_f'^{-1}$ pour se débarrasser du déchet :

$$\begin{array}{rcl}
|x\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes k} \otimes |y\rangle & \xrightarrow{U'_f} & |x\rangle \otimes |f(x)\rangle \otimes |g(x, 0^m)\rangle \otimes |y\rangle \\
& \xrightarrow{C - NOT^{\otimes m}} & |x\rangle \otimes |f(x)\rangle \otimes |g(x, 0^m)\rangle \otimes |y \oplus f(x)\rangle \\
& \xrightarrow{U_f'^{-1}} & |x\rangle \otimes |0\rangle^{\otimes m} \otimes |0\rangle^{\otimes k} \otimes |y \oplus f(x)\rangle
\end{array}$$

[10] On a en particulier $T|a, b, 0\rangle = |a, b, a \wedge b\rangle$ et $T|1, 1, c\rangle = |1, 1, \bar{c}\rangle$, ce qui permet de simuler les portes classiques AND et NOT.

[11] La copie d'état quantique est impossible, mais ici on se contente de copier les états de base, ce qui ne pose pas de problème.

On a donc bien éliminé le déchet, et en oubliant les qubits dans l'état $|0\rangle$, on obtient bien U_f.

Remarquons que $U_f(|x\rangle|0\rangle^m) = |x\rangle|f(x)\rangle$; il est donc toujours possible d'accéder à $f(x)$, à condition d'avoir m registres supplémentaires dans l'état 0 pour stocker la réponse. De plus, la porte U_f ainsi définie est toujours unitaire quelque soit f.

Par ailleurs, lorsque $m = 1$ (donc lorsque f est une fonction booléenne), on utilise souvent à la place de U_f la porte S_f, qui agit sur n qubits seulement (contre $n + 1$ pour U_f) de la manière suivante :

$$S_f|x\rangle = (-1)^{f(x)}|x\rangle$$

c'est-à-dire que S_f effectue une inversion globale de phase sur $|x\rangle$ si et seulement $f(x) = 1$. Cette nouvelle forme de requête peut être obtenue à partir de U_f de la manière suivante[12] : si on note $|\psi\rangle = |0\rangle - |1\rangle$, alors on a la relation : $U_f\big(|x\rangle|\psi\rangle\big) = \big(S_f|x\rangle\big)|\psi\rangle$. On pourra donc utiliser l'une ou l'autre de ces définitions suivant les besoins.

1.2.5 Famille uniforme de circuits

Un circuit donné a un nombre fixé de qubits en entrée et ne peut donc pas résoudre un problème de manière générale, comme par exemple le tri d'une liste de taille indéterminée. Pour ce faire, on est obligé de considérer une famille infinie de circuits dont la taille de l'entrée varie. Mais comme en calcul classique, autoriser n'importe quelle famille de circuits se révèle irréaliste et trop puissant ; on est donc amené à imposer des limites raisonnables.

C'est pour cette raison que l'on définit la notion de *famille uniforme* de circuits, de la manière suivante : une famille de circuits $(C_n)_{n\in\mathbb{N}}$ est uniforme si et seulement si C_n peut être décrit par une machine de Turing classique déterministe sur l'entrée n en temps polynomial. Cette "description" doit comporter non seulement la structure du circuit, mais aussi les portes utilisées, dont les caractéristiques, en particulier les coefficients complexes, doivent donc pouvoir être décrites efficacement de manière algorithmique.

Cette définition achève la description du modèle de circuits quantiques ; à présent, nous allons nous intéresser à une classe spéciale de problèmes dans ce modèle, les problèmes avec oracle, et définir la notion de complexité que nous utiliserons par la suite.

Pour faire une synthèse de ce que nous avons vu sur le modèle de calcul, si l'on considère un problème $P : \bigcup_{n=0}^{\infty}\{0,1\}^n \to \{0,1\}$, la complexité de P est le nombre minimum de portes nécessaires, en fonction de n, pour calculer $P_{|\{0,1\}^n}$ à l'aide d'une famille uniforme de circuit avec une erreur $\epsilon < 1/2$ donnée ; changer ϵ ne modifie que la constante multiplicative[13], et n'a donc pas de véritable incidence sur la complexité.

1.3 Modèle de la boîte noire

Le problème du modèle de calcul ainsi défini est qu'il n'existe jusqu'à présent pratiquement aucune borne inférieure de complexité, ce qui laisse penser que la découverte de telles bornes est un problème difficile. Pour remédier à cela, on est obliger d'introduire des contraintes ou des hypothèses supplémentaires sur ce modèle. On introduit ainsi le modèle de la *boîte noire*.

[12]La réciproque (c'est-à-dire définir U_f à partir de S_f) est vraie également, mais à condition que S_f soit donné par un circuit, ou de manière plus générale que l'on puisse réaliser un $CONTROLLED - S_f$.

[13]Il suffit d'effectuer un vote par majorité en utilisant le circuit plusieurs fois consécutives ; pour répéter un algorithme quantique, il suffit de l'effectuer, de "copier" le résultat sur un qubit de travail dans l'état $|0\rangle$ avec un $CONTROLLED - NOT$, de faire tourner l'algorithme à l'envers pour revenir à l'état initial (sauf en ce qui concerne le qubit de travail), et itérer.

Dans ce modèle, également utilisé en informatique classique, on a accès à un oracle, ou boîte noire, qui est une fonction $f : [1, N] \to [1, M]$[14]. L'objectif est de déterminer si f vérifie une certaine propriété P (par exemple : "Est-ce que f prend une certaine valeur j ?") en soumettant à cet oracle des requêtes du type : "Quel est la valeur de f en i ?". La complexité associée à un algorithme, et que l'on cherche donc à minimiser, est alors le nombre de requêtes effectuées pour résoudre le problème.

En calcul quantique, le modèle de la boîte noire est quasiment identique, à ceci près qu'au lieu de demander la valeur de f en un point, on utilise l'équivalent quantique de cette demande, c'est-à-dire la porte U_f définie précédemment dans le paragraphe 1.2.4. De manière similaire, la complexité d'un circuit C résolvant ce problème pour une instance donnée de f est le nombre d'appels à U_f, et au lieu de dépendre de la taille de l'entrée de ce circuit, elle dépend de manière naturelle de la taille de l'oracle (c'est-à-dire de N et M). C'est ce que l'on appellera par la suite le *modèle d'évaluation* ; en effet, chaque requête effectuée demande en fait d'évaluer f en un point ou une superposition quantique de points.

En pratique, le calcul effectué par C commence alors avec l'état $|0\rangle$, et s'achève sur la mesure d'un certain qubit, par exemple celui situé le plus à droite. C calcule alors P avec erreur bornée, si pour tout oracle f, le résultat de la mesure finale est $P(f)$ avec probabilité au moins $1 - \epsilon$, pour un certain $\epsilon < 1/2$.

2 Outils utilisés

Cette partie présente quelques résultats dont nous allons avoir besoin plus tard : l'algorithme de recherche de Grover, dont l'utilisation est essentielle dans les algorithmes donnés par la suite, et deux méthodes très différentes pour établir des bornes inférieures de complexité.

2.1 Algorithme de Grover

L'algorithme de Grover est un résultat fondamental de calcul quantique[8], permettant de résoudre plus efficacement qu'en calcul classique le problème de recherche dans un ensemble non trié. La présentation de l'algorithme qui suit est inspirée de Høyer[10].

2.1.1 Description du problème

On se donne un oracle $f : [1, N] \to \{0, 1\}$ (on suppose $N = 2^n$, c'est-à-dire que l'on travaille sur n qubits). On appelle solution du problème de recherche une valeur x d'entrée telle que $f(x) = 1$; on suppose que f admet t solutions et que t est connu. Classiquement, comme on n'a aucune autre information que l'existence de solutions, cela se fait de manière optimale en sondant l'entrée jusqu'à trouver une solution ; si l'on s'intéresse à un modèle probabiliste, cela donne une complexité de $\Theta(N/t)$. En calcul quantique, nous allons voir que l'algorithme de Grover permet d'obtenir une complexité en $\Theta(\sqrt{N/t})$.

2.1.2 Amplification d'amplitude

Le lemme suivant est fondamental ; comme son nom l'indique, il permet d'amplifier l'amplitude de certains états (ici, ceux dont le dernier qubit est dans l'état $|1\rangle$). Le point important est que cette amplification est quasi-linéaire ; or, en calcul quantique, l'amplitude est égale à la racine carré

[14]Ici, et dans le reste de ce rapport, on note $[1, N] = \{1, \ldots, N\}$ et plus généralement $[a, b] = \{a, a+1, \ldots, b\}$

de la probabilité. En calcul classique, on sait seulement augmenter la probabilité de réussite ; en calcul quantique, il est donc possible d'augmenter la racine carré de la probabilité de réussite, et c'est pour cette raison que l'on obtient finalement un gain quadratique.

Lemme 2.1 (Amplification d'Amplitude) *On considère un algorithme A agissant sur n qubits, et tel que :*

$$A(|0\rangle^{\otimes n}) = \sin(\theta)|\phi_1\rangle|1\rangle + \cos(\theta)|\phi_0\rangle|0\rangle$$

avec $|\phi_0\rangle$ et $|\phi_1\rangle$ deux vecteurs quelconques normés sur $n-1$ qubits. On note S_0 et S_1 les opérateurs agissant sur n qubits définis par :

$$\begin{cases} S_0(|0\rangle^{\otimes n}) & = & -|0\rangle^{\otimes n} \\ S_0|\phi\rangle & = & |\phi\rangle \qquad \forall\ |\phi\rangle \neq |0\rangle \end{cases}$$

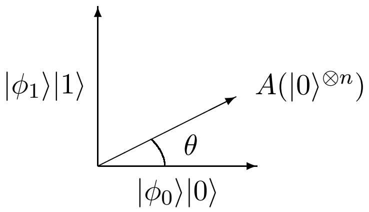
$$\forall\ |\psi\rangle \in H_{2^{n-1}}, \begin{cases} S_1(|\psi\rangle \otimes |1\rangle) & = & -|\psi\rangle \otimes |1\rangle \\ S_1(|\psi\rangle \otimes |0\rangle) & = & |\psi\rangle \otimes |0\rangle \end{cases}$$

c'est-à-dire que S_0 change la phase de $|0\rangle^{\otimes n}$ et laisse toute autre entrée inchangée, tandis que S_1 change la phase de toute entrée dont le dernier qubit est dans l'état $|1\rangle$. Si l'on pose alors $G_A = -AS_0A^{-1}S_1$, on a le résultat suivant :

$$G_A A(|0\rangle^{\otimes n}) = \sin(3\theta)|\phi_1\rangle|1\rangle + \cos(3\theta)|\phi_0\rangle|0\rangle$$

Dans l'énoncé de ce lemme, les états dont le dernier qubit est à $|1\rangle$ représentent les états de succès de l'algorithme A, et $\sin^2(\theta)$ est donc la probabilité d'un tel succès.

Outre d'amplifier l'amplitude des états dont le dernier qubit est à $|1\rangle$, que fait ce lemme de manière intuitive ? La forme donnée à $A(|0\rangle^{\otimes n})$ fait bien entendu penser à une interprétation géométrique ; en effet, si l'on se place dans le sous-espace vectoriel de dimension 2 généré par $|\phi_1\rangle|1\rangle$ et $|\phi_0\rangle|0\rangle$, alors on a le schéma suivant :



L'application du lemme va donc changer l'angle θ en 3θ, rapprochant l'état du système, à savoir $A(|0\rangle^{\otimes n})$, d'un état de succès, c'est-à-dire $|\phi_1\rangle|1\rangle$. Cette interprétation permet par ailleurs d'avoir une idée de la preuve du lemme ; en effet, les opérations appliquées lors de son utilisation sont les suivantes :

1. S_1, c'est-à-dire une réflexion par rapport à $|\phi_0\rangle|0\rangle$;
2. $-AS_0A^{-1} = A(-S_0)A^{-1}$, qui est la conjuguée par A d'une réflexion par rapport à $|0\rangle^{\otimes n}$; il s'agit donc en fait d'une réflexion par rapport à $A|0\rangle^{\otimes n}$.

Ainsi, l'opérateur G_A est en fait une composition de deux réflexions ; il s'agit donc finalement d'une rotation d'angle deux fois l'angle entre les axes de ces réflexions, c'est-à-dire ici θ. Comme G_A est une rotation d'angle 2θ, on obtient bien le résultat enoncé par le lemme.

2.1.3 Présentation de l'algorithme

Pour alléger les notations, on notera $|0\rangle = |0\rangle^{\otimes n}$.

On procède par récurrence en utilisant le lemme d'amplification d'amplitude. Voyons d'abord comment amorcer cette récurrence. Comme suggéré dans la preuve du lemme précédent, nous effectuons une séparation entre les états de succès ou d'échec du problème de recherche. Pour obtenir la séparation suivant l'état du dernier qubit, il suffit d'appliquer U_f ; en effet $U_f|x\rangle|0\rangle = |x\rangle|1\rangle$ si et seulement si $f(x) = 1$. Par ailleurs, si l'on veut que l'amplification ait un effet, il faut être sûr d'avoir au moins un bon état. Comme on ne connaît rien sur f, on utilise une superposition uniforme de tous les états possibles ; cette superposition peut être obtenue, on le rappelle, en utilisant la porte d'Hadamard en parallèle sur des qubits dans l'état $|0\rangle$.

Par conséquent, on définit $A_0 = U_f H^{\otimes n}$; appliqué à $|0\rangle$, on obtient la superposition suivante :

$$A_0|0\rangle = \frac{1}{\sqrt{2^n}}\left(\left(\sum_{f(x)=1}|x\rangle\right)|1\rangle + \left(\sum_{f(x)=0}|x\rangle\right)|0\rangle\right)$$

C'est-à-dire

$$A_0|0\rangle = \sin(\theta_0)|\phi_1\rangle|1\rangle + \cos(\theta_0)|\phi_0\rangle|0\rangle$$

avec (rappelons que t est le nombre de x tels que $f(x) = 1$ et que $N = 2^n$ est la taille de l'espace de recherche) :

$$\sin(\theta_0) = \sqrt{\frac{t}{N}} \quad \text{et} \quad \cos(\theta_0) = \sqrt{\frac{N-t}{N}}$$

$$|\phi_1\rangle = \frac{1}{\sqrt{t}}\left(\sum_{f(x)=1}|x\rangle\right)$$

$$|\phi_0\rangle = \frac{1}{\sqrt{N-t}}\left(\sum_{f(x)=0}|x\rangle\right)$$

Ici, θ_0 est choisi de manière à ce que $|\phi_1\rangle$ et $|\phi_0\rangle$ soit normés.

On peut donc appliquer le lemme à A_0 (l'inverse de A_0 est facile à construire, puisque $U_f^{-1} = U_f$ et $H^{-1} = H$; il en sera de même pour tous les A_k) ; on construit ainsi $A_1 = G_{A_0}A_0$. $A_1|0\rangle$ est alors exactement de la même forme que $A_0|0\rangle$ en remplaçant θ_0 par $\theta_1 = 3\theta_0$. Ainsi, par récurrence, on définit $A_{k+1} = G_{A_k}A_k$, avec à chaque fois $\theta_{k+1} = 3\theta_k$; on a donc $\theta_k = 3^k\theta_0$. La question est maintenant de savoir quand s'arrêter. Pour ce faire, il suffit de remarquer que $\sin(\theta_0)$ est petit, car on peut supposer le nombre t de solutions petit par rapport à N ; dans le cas contraire, la recherche d'une solution se fait déjà en complexité constante. Par conséquent :

$$\theta_0 \approx \sin(\theta_0) \approx \sqrt{\frac{t}{N}}$$

$$\theta_k \approx 3^k\sqrt{\frac{t}{N}}$$

Pour avoir une probabilité de succès maximale, il faut avoir $\sin(\theta_k)$ proche de 1, c'est-à-dire θ_k proche de $\pi/2$. On prend donc :

$$k = \left\lfloor \log_3\left(\frac{\pi}{2}\sqrt{\frac{N}{t}}\right)\right\rceil$$

En effectuant une mesure, on obtient alors avec grande probabilité une solution. Nous avons donc décrit comment construire par récurrence un circuit résolvant le problème de recherche.

L'analyse de la complexité de cet algorithme est très rapide ; en effet, si on note C_k le nombre d'appels à f effectués par A_k, on a $C_0 = 1$ et $C_k = 3C_{k-1}$. Par conséquent :

$$C_k = 3^k \approx \frac{\pi}{2}\sqrt{\frac{N}{t}}$$

ce qui est bien le résultat recherché. Cette complexité est de plus optimale, comme le verrons en exemple de borne inférieure utilisant la méthode d'Ambainis.

2.1.4 Conclusion

L'algorithme de Grover permet donc une accélération quadratique par rapport au calcul classique. Bien que cela n'est a priori pas aussi impressionnant que les gains exponentiels promis par l'algorithme de Shor, ce résultat n'est pas sans intérêt. Ainsi, contrairement à la factorisation, le gain ici, bien que seulement polynomial, est garanti ; dans le cas de l'algorithme de Shor, bien que ce soit probable, il n'a pas encore été montré qu'un algorithme classique de factorisation était nécessairement exponentiel[15]. De plus, comme nous le verrons par la suite, autour du problème de recherche existent un certain nombre de problèmes annexes dont on n'a pas encore cerné exactement les complexités.

Remarquons également que le lemme d'amplification d'amplitude, comme suggéré dans son introduction, peut en fait s'appliquer à n'importe quel algorithme dont la probabilité p de succès est mise sous la forme d'un $\sin^2(\theta)$; en effectuant une analyse similaire à celle présentée ici, on voit donc qu'il suffit de $\frac{1}{\sqrt{p}}$ itérations de cet algorithme pour obtenir une grande probabilité de succès, alors que l'on a besoin de $\frac{1}{p}$ itérations en calcul classique pour obtenir le même résultat. On retrouve donc à nouveau le gain quadratique dû au modèle de calcul quantique.

Par ailleurs, cette présentation suppose que l'on connaisse le nombre de solutions, ou la probabilité p de succès. Il est en fait possible de créer une variante de cet algorithme qui fonctionne même si l'on ne connaît pas le nombre de solutions, avec la même complexité à une constante multiplicative près.

Mentionnons enfin le résultat démontré par Høyer mentionné en introduction de cette partie :

Théorème 2.2 *[10] Si on autorise l'oracle à répondre avec erreur bornée, alors la complexité du problème de recherche reste inchangée ($\Theta(\sqrt{N/t})$).*

L'algorithme présenté pour montrer ce théorème a la même structure que celui décrit dans ce rapport, mais une correction d'erreur, par vote majorité sur un nombre croissant (de manière linéaire) d'appels à l'oracle, est ajoutée à chaque étape de récursion ; les calculs deviennent alors beaucoup plus compliqués, mais similaires dans le principe.

2.2 Bornes inférieures par la méthode par adversaire

L'objectif de cette partie est de présenter la méthode d'Ambainis[2], dite d'adversaire quantique, pour déterminer des bornes inférieures de complexité.

[15] L'absence d'une telle borne inférieure est cependant sans doute due au fait que ce problème n'est pas dans le modèle de la boîte noire. Le Hidden-Subgroup Problem, qui est exprimé dans ce modèle, et à la base de l'algorithme de Shor, possède lui une borne inférieure exponentielle en calcul classique.

2.2.1 Notations

On se donne donc un problème, c'est-à-dire une fonction $P : \{0,1\}^{[1,N]} \to [1,n]$, qui associe à toute fonction booléenne $f : [1,N] \to \{0,1\}$ une certaine valeur dans $[1,n]$. Dans le cas particulier où le domaine de P est de cardinal 2, on le notera plutôt $\{0,1\}$: dans ce cas, on dit que P est un problème de décision.

Un circuit quantique C utilisant T appels à l'oracle f est alors de la forme :

$$U_0 \to O \to U_1 \to O \to \ldots \to U_{T_1} \to O \to U_T$$

où les U_j sont des transformations unitaires qui ne dépendent pas de f, tandis que O représente un appel à f ; O associe ainsi à l'état de base $|i,z\rangle$ l'état $(-1)^{f(i)}|i,z\rangle$[16]. i est donc la valeur en laquelle on effectue une évaluation de f (et est donc constitué de $\lceil \log N \rceil$ qubits), et z contient tous les autres qubits du circuit.

2.2.2 Adversaire quantique

La méthode de l'adversaire consiste à présenter à un algorithme résolvant le problème étudié des entrées f et g, dont le résultat final est différent (donc $P(f) \neq P(g)$), et considérer ensuite une certaine quantité S_k, définie à chaque étape de calcul k (et dépendant de f et de g), telle que :

$$S_0 - S_T \geq M$$

$$\forall k,\ S_k - S_{k+1} \leq L$$

c'est-à-dire que S varie beaucoup sur l'ensemble de l'algorithme, mais peu à chaque étape de calcul. Ces 3 conditions impliquent de manière évidente une borne inférieure sur le nombre T d'appels à l'oracle :

$$T \geq \frac{M}{L}$$

2.2.3 Borne inférieure

Nous commençons par énoncer sans le démontrer un lemme qui nous permettra de séparer des états finals de l'algorithme produisant des résultats différents. La preuve de ce lemme est directe, mais essentiellement trigonométrique, et donc pénible.

Lemme 2.3 *Soit deux états quantiques $|\phi_1\rangle$ et $|\phi_2\rangle$. On considère alors un état $|\phi\rangle = |\phi_i\rangle$ avec $i \in \{1,2\}$ complètement inconnu. Alors il existe une mesure sur $|\phi_i\rangle$ permettant de déterminer i avec probabilité d'erreur au plus ϵ si et seulement si $|\langle\phi_1|\phi_2\rangle| \leq 2\sqrt{\epsilon(1-\epsilon)}$.*

Le résultat principal est le suivant :

Théorème 2.4 *On considère X et Y deux ensembles d'entrées tels que $P(f) \neq P(g)$ dès que $f \in X$ et $g \in Y$. Soit $R \subset X \times Y$ telle que :*

1. *pour tout $f \in X$, il y a au moins m différents $g \in Y$ tels que $(f,g) \in R$;*
2. *pour tout $g \in Y$, il y a au moins m' différents $f \in X$ tels que $(f,g) \in R$;*
3. *pour tout $f \in X$ et $i \in [1,N]$, il y a au plus l différents $g \in Y$ tels que $(f,g) \in R$ et $f(i) \neq g(i)$;*

[16] On utilise des requêtes du type S_f et non U_f.

4. pour tout $g \in Y$ et $i \in [1,N]$, il y a au plus l' différents $f \in X$ tels que $(f,g) \in R$ et $f(i) \neq g(i)$;

Alors, tout algorithme quantique calculant P utilise $\Omega(\sqrt{\frac{mm'}{ll'}})$ appels à l'oracle.

Remarquons que l'équivalent classique probabiliste de ce théorème permet seulement d'obtenir une borne inférieure en $\Omega\left(\max\left(\frac{m}{l}, \frac{m'}{l'}\right)\right)$.

Les deux premières conditions permettent globalement de minorer $S_0 - S_k$; en effet, la mesure S que l'on va définir dans la preuve dépend du nombres d'éléments de R, lequel peut être minoré en fonction de m et m'. Réciproquement, les deux dernières servent à majorer $S_k - S_{k+1}$. La condition $f(i) \neq g(i)$ vient du fait que sur tous les autres points, où f et g sont donc égaux, l'oracle va agir de la même manière et ne peut donc pas influer sur S. L'intérêt de ce théorème est qu'il offre un cadre très général pour montrer des bornes inférieures de complexité, cadre qui peut s'appliquer à de nombreux problèmes, comme nous le verrons par la suite ; la difficulté de son utilisation réside dans le choix des paramètres X, Y et R.

La preuve complète de ce théorème est donnée en annexe.

2.2.4 Généralisation

Le précédent théorème représente en fait la version allégée de la méthode d'Ambainis. Le résultat principal, dont nous ne donnerons pas la démonstration (laquelle est très semblable à la preuve du théorème 2.4), est le suivant :

Théorème 2.5 *Soit $P : [1,M]^{[1,N]} \to [1,n]$ un problème associant à un oracle $f : [1,N] \to [1,M]$ une certaine valeur $P(f)$. On considère X et Y deux ensembles d'entrées tels que $P(f) \neq P(g)$ dès que $f \in X$ et $g \in Y$. Soit $R \subset X \times Y$ telle que :*

1. pour tout $f \in X$, il y a au moins m différents $g \in Y$ tels que $(f,g) \in R$;

2. pour tout $g \in Y$, il y a au moins m' différents $f \in X$ tels que $(f,g) \in R$;

Pour tout $f \in X$ et tout $i \in [1,N]$, on définit $l_{f,i}$ le nombre de $g \in Y$ tels que $(f,g) \in R$ et $f(i) \neq g(i)$; on définit symétriquement $l'_{g,i}$ pour tout $g \in Y$ et tout i. On pose alors :

$$l_{\max} = \max_{(f,g)\in R, i : f(i)\neq g(i)} l_{f,i} l'_{g,i}$$

Alors, tout algorithme quantique calculant P utilise $\Omega(\sqrt{\frac{mm'}{l_{\max}}})$ appels à l'oracle.

Ce théorème appelle deux commentaires. D'une part, on autorise n'importe quelle fonction en oracle, et non plus seulement des fonctions booléennes comme précédemment. D'autre part, le produit ll' est remplacé par $l_{\max}$; or dans le théorème précédent, on a en fait $l = \max_{f,i} l_{f,i}$ et $l' = \max_{g,i} l'_{g,i}$. Il est donc facile de voir que $l_{\max} \leq ll'$. Par conséquent, cette nouvelle borne inférieure est au moins aussi bonne que celle donnée précédemment.

2.2.5 Exemple : le problème de recherche

On considère tout d'abord le problème de décision associé ; c'est-à-dire que l'on se donne $f : [1,N] \to \{0,1\}$, et l'on veut savoir s'il existe s tel que $f(x) = 1$, avec la promesse qu'il y a exactement 0 ou t solutions.

Notons ici que la complexité de ce nouveau problème est égale à celle du problème de recherche, avec au pire un facteur logarithmique supplémentaire ; en effet, la réduction du problème de recherche au problème de décision associé se fait par dichotomie. Réciproquement, on peut réduire le problème de décision au problème de recherche, sans perte de complexité, de la manière suivante : si l'on se donne un algorithme $A_{\text{recherche}}$ résolvant le problème de recherche, on peut en déduire immédiatement un algorithme $A_{\text{décision}}$ de même complexité résolvant le problème de décision de la manière suivante :

1. Appliquer $A_{\text{recherche}}$ à f pour obtenir un résultat x.
2. Si $f(x) = 1$ alors répondre oui.
3. Sinon répondre non.

En effet, si f a effectivement une solution, $A_{\text{recherche}}$ va la trouver ; dans le cas contraire, $A_{\text{recherche}}$ renvoie n'importe quoi. Par conséquent, la borne inférieure que nous allons maintenant montrer pour le problème de décision va être également valable pour le problème de recherche.

On pose alors :

$$X = \{f \mid f \text{ n'a pas de solution.}\}$$

$$Y = \{g \mid g \text{ a exactement } t \text{ solutions.}\}$$

$$(f, g) \in R \text{ si et seulement si } f \text{ et } g \text{ diffèrent en } t \text{ points.}$$

Une remarque importante ici est que la donnée de R est strictement équivalente à la donnée de l'une des deux fonctions $R_X : X \to \mathcal{P}(Y)$ et $R_Y : Y \to \mathcal{P}(X)$ définies par :

$$\forall f \in X,\ R_X(f) = \{g \mid (f, g) \in R\}$$

et vice-versa pour R_Y.

L'intérêt de cette démarche est qu'il suffit alors de donner X et R_X pour en déduire $Y = R_X(X)$ et $R = \{(f, g) \mid f \in X \text{ et } g \in R_X(X)\}$. Et plutôt que de définir R_X formellement, on se contentera d'en donner une description. Par exemple, dans le cas qui nous occupe actuellement, on dira :

$$R_X(f) = \text{"On change exactement } t \text{ valeurs de } f.\text{"}$$

Il pourra parfois être utile d'expliciter R_Y, ceci afin de simplifier les calculs qui s'ensuivent. Ici, on a :

$$R_Y(g) = \text{"On supprime toutes les solutions de } g.\text{"}$$

On calcule maintenant les coefficients du théorème 2.4 (dans ce qui suit, f et g représentent respectivement des éléments de X et Y) :

- $m = \dfrac{N!}{t!(N-t)!}$ car pour passer de f qui n'a pas de solution à g avec t solutions, il faut choisir ces t solutions parmi N possibilités ;
- $m' = 1$, car pour supprimer les t solutions de g en changeant seulement t valeurs, aucun choix n'est possible ;
- $l = \dfrac{(N-1)!}{(t-1)!(N-t)!}$, car on a fixé un point i où $f(i) \neq g(i)$, c'est-à-dire que l'on *force* f à changer en ce point i. Par conséquent, il n'y a que $t-1$ autres changements parmi $N-1$ à choisir ;
- $l' = 1$, car quoiqu'il arrive, il n'y a toujours au plus qu'une seule manière de passer de g à f.

Cela nous donne donc une borne inférieure de $\Omega\left(\sqrt{\frac{N}{t}}\right)$. Le problème de recherche dans une liste non triée a donc bien une complexité de $\Theta\left(\sqrt{\frac{N}{t}}\right)$.

Nous verrons de nombreux autres exemples d'utilisation de ce théorème (et en particulier de sa généralisation) dans la dernière partie de ce rapport.

2.3 Bornes inférieures par la méthode des polynômes

2.3.1 Principe

Nous allons maintenant présenter une deuxième méthode, très différente et due à Beals, Buhrman, Cleve, Mosca et de Wolf[4], pour obtenir d'autres bornes inférieures. Comme son nom l'indique, l'idée de cette méthode est d'utiliser des polynômes. Plus précisément, on procède en trois étapes :

1. Établir une relation entre la complexité d'un circuit quantique et le degré d'un polynôme en de nombreuses variables dépendant de l'oracle du circuit.
2. Symétriser ce polynôme pour réduire le nombre de variable à une constante, en se limitant à une certaine classe d'oracles ; cette étape peut être très différente suivant les problèmes à résoudre (et c'est d'ailleurs la partie difficile de l'utilisation de la méthode), ce pourquoi on ne la détaillera pas plus dans l'immédiat.
3. Minorer le degré du polynôme ainsi obtenu, en utilisant des contraintes (dues au fait qu'il s'agit d'un polynôme représentant la probabilité de succès d'un algorithme), et avec des résultats mathématiques d'approximation. Grâce aux deux étapes précédentes, cette minoration s'étend alors à la complexité du circuit.

2.3.2 Polynôme associé à un circuit C

Comme décrit précédemment, un circuit C effectuant T requêtes à l'oracle est une succession de portes de la forme :

$$U_0 \to O \to U_1 \to O \to \ldots \to U_{T_1} \to O \to U_T$$

On définit à partir de f les variables $\delta_{j,i}$ suivantes[17] :

$$\forall i \in [1, N], \forall j \in [1, M],\ \delta_{j,i}(f) = 1 \Longleftrightarrow f(i) = j$$

On a alors le lemme suivant :

Lemme 2.6 *Il existe des polynômes α_z à variables $\delta_{j,i}$, de degré au plus T, tel que l'état final de C est de la forme :*

$$\sum_z \alpha_z(f)|z\rangle$$

Pour montrer ce lemme, on procède par récurrence ; on note $|\psi_f^k\rangle$ l'état de l'algorithme après k étapes de calcul avec l'oracle f. Alors $|\psi_f^0\rangle$ ne dépend pas de f, et ses amplitudes sont donc des constantes, c'est-à-dire des polynômes de degré 0. Ensuite, si $|\psi_f^k\rangle = \sum_z \alpha_z^k(f)|z\rangle$ avec $\alpha_z^k(f)$ de degré k, les amplitudes de $U_k|\psi_f^k\rangle$ sont des combinaisons linéaires de polynômes de degré k et

[17] Ici on échange les indices i et j par rapport à la notation usuelle, ceci afin d'avoir les entrées de f en abscisse et les valeurs prises par f en ordonnée lorsque l'on considérera la matrice $(\delta_{j,i})_{1\leq j\leq M, 1\leq i\leq N}$

restent donc de degré k. On peut donc virtuellement oublier les U_k et il nous reste donc à montrer que le degré des $\alpha_z(f)$ augmente d'au plus 1 à chaque appel à f. En effet, en réécrivant $|\psi_f^k\rangle$ dans la base de requête, on a :

$$|\psi_f^k\rangle = \sum_{i,b,z'} \alpha_{i,b,z'}^k(f)|i,b,z'\rangle \longrightarrow O|\psi_f^k\rangle = \sum_{i,b,z'} \alpha_{i,b,z'}^k(f)|i,b \oplus f(i),z'\rangle$$

En effectuant un changement de variable, on obtient :

$$O|\psi_f^k\rangle = \sum_{i,b,z'} \alpha_{i,b\oplus f(i),z'}^k(f)|i,b,z'\rangle$$

On a donc :

$$\alpha_{i,b,z'}^{k+1}(f) = \alpha_{i,b\oplus f(i),z'}^k(f) = \sum_{1\leq j\leq M} \delta_{j,i}(f)\alpha_{i,b\oplus j,z'}^k(f)$$

qui est bien un polynôme de degré au plus $k+1$. □

On en déduit immédiatement le corollaire suivant :

Corollaire 2.7 *La probabilité de mesurer un résultat donné à la fin du circuit C est un polynôme en les variables $\delta_{j,i}$ et de degré au plus $2T$ (le facteur 2 provient de l'élévation au carré de l'amplitude).*

2.3.3 Lemme d'approximation

La dernière étape de la méthode des polynômes implique de savoir minorer le degré d'un polynôme ayant des contraintes fortes. Pour ce faire, on utilisera le résultat suivant, donné sans preuve ; cette dernière est difficile et basée sur des polynômes trigonométriques :

Théorème 2.8 (Paturi) *[13, 14] Soit $q(X)$ un polynôme à une variable de degré d. Soit a et b des entiers ($a < b$), $\xi \in [a,b]$ un réel et c_1, c_2 deux constantes réelles strictement positives. On suppose que :*

1. *$|q(i)| \leq c_1$ pour tout entier $i \in [a,b]$;*
2. *$|q(\lfloor\xi\rfloor) - q(\xi)| \geq c_2$*

Alors :

$$d = \Omega\left(\sqrt{(\xi - a + 1)(b - \xi + 1)}\right)$$

Intuitivement, pour obtenir la meilleure borne inférieure possible, on va donc chercher à obtenir des polynômes bornés sur les valeurs entières d'un intervalle le plus large possible, et possédant cependant un saut à l'intérieur de cet intervalle. Remarquons deux cas particuliers importants de ce théorème :

- lorsque $\xi \approx a$ ou $\xi \approx b$, $d = \Omega(\sqrt{b-a})$;
- et lorsque $\xi \approx \frac{a+b}{2}$, $d = \Omega(b-a)$.

On essaiera donc de préférence de mettre le saut vers le milieu de l'intervalle considéré.

Ceci achève la présentation de cette méthode des polynômes. Bien entendu, cela ne donne pas vraiment l'idée des détails de son fonctionnement, notamment en ce qui concerne la deuxième étape ; c'est pourquoi nous verrons au cours de la troisième et dernière partie de ce rapport un exemple d'utilisation.

3 Problèmes étudiés

Introduction : Modèle de Test

Cette section présente l'étude de la complexité d'un certain nombre de problèmes étudiés pendant le stage. Mais avant de commencer nous allons commencer par faire une observation importante, concernant la méthode des polynômes que nous venons de décrire. Revenons donc sur la première étape de cette méthode, à savoir la définition des polynômes d'amplitudes associés à un circuit C ; les polynômes obtenus sont de la forme :

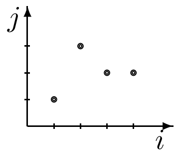
$$\sum_{1 \le j \le M} \delta_{j,i}(f)\alpha^k_{i,b\oplus j,z'}(f)$$

Par l'intermédiaire de $\delta_{j,i}$, chacun des termes de cette somme représente en quelque sorte un *test*, permettant de déterminer si $j = f(i)$, et ajustant le polynôme représentant l'amplitude en conséquence. Mais le point intéressant ici est qu'*un seul* de ces termes suffit en fait à augmenter le degré de ce polynôme ; en effet, si l'on supprime le symbole de somme dans la formule précédente (pour obtenir $\delta_{j,i}(f)\alpha^k_{i,b\oplus j,z'}(f)$ pour un certain couple (j, i)), le polynôme résultant est de même degré que celui formé de la somme de tous ces termes. On est donc naturellement amené à considérer un modèle d'oracle beaucoup plus faible que le modèle d'évaluation utilisé jusqu'à maintenant. Les requêtes, au lieu d'être comme jusqu'à présent de la forme $i \to f(i)$, seront dans ce nouveau modèle de la forme plus faible suivante :

$$(j, i) \to \begin{cases} \text{1 si j=f(i)} \\ \text{0 sinon} \end{cases}$$

Ce modèle sera appelé *modèle de test*. De manière équivalente, on considère dans le modèle d'évaluation une matrice $\delta_{j,i}$ à valeurs booléennes, telle qu'il y ait exactement un 1 par colonne ; chaque colonne correspond ainsi à une élément de l'ensemble de départ de f. Par exemple :

$$\delta = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \Longleftrightarrow$$



Dans ce modèle pourtant a priori plus faible (puisqu'on ne peut avoir la valeur de $f(i)$ facilement, à moins d'avoir de la chance), la méthode des polynômes ne peut pourtant pas donner pour un certain problème de meilleures bornes inférieures que pour le modèle d'évaluation :

Remarque 3.1 *Toute borne inférieure montrée pour un problème dans le modèle de test par la méthode des polynômes est également valable pour ce même problème dans le modèle d'évaluation.*

On considère la preuve de la borne inférieure par la méthode des polynômes pour le modèle de test, et l'on va voir qu'elle s'applique également au modèle d'évaluation :

1. le polynôme associé à n'importe quel circuit C dans le modèle d'évaluation utilise les mêmes variables, et est de degré borné par la même quantité, à savoir deux fois le nombre de requêtes T du circuit, que dans le modèle de test ; ceci est dû à l'observation faite plus haut ayant mené à la définition du modèle de test ;
2. la symétrisation ne dépend pas du modèle de requête mais seulement de la structure même du problème que l'on cherche à résoudre ; comme ce problème est identique dans les deux modèles, cette étape peut être reproduite à l'identique dans le modèle d'évaluation ;

3. la minoration du degré du polynôme ainsi obtenu se fait en utilisant des contraintes dues au problème à résoudre ; là encore, le fait d'être dans le modèle de test ou dans le modèle d'évaluation n'intervient pas. □

Remarque 3.2 *En fait cette remarque ne s'applique pas seulement au modèle de test, mais également à un modèle probabiliste classique ; en effet, la seule différence dans la preuve de borne inférieure par la méthode des polynômes dans ce modèle classique est que le degré du polynôme représentant la probabilité de succès est borné non pas par $2T$ mais seulement par T, ce qui ne change pas l'ordre de grandeur de la borne inférieure obtenue à la fin.*

Ce théorème montre bien une limite fondamentale de cette méthode des polynômes. Une question naturelle maintenant est de savoir s'il existe des problèmes plus difficiles dans le modèle de test que dans le modèle d'évaluation ; la réponse est oui, comme nous allons le voir par la suite, ce qui montre bien une limite de la méthode des polynômes.

De manière plus générale, on s'interroge naturellement sur la puissance comparée de ces deux modèles ; comme un test peut être évidemment simulée par une évaluation, le modèle de test est plus faible. Réciproquement, on remarque qu'il est possible, grâce à l'algorithme de Grover, d'évaluer la valeur de f en un point avec $\sqrt{M}$ tests. En effet, il s'agit tout simplement de la recherche d'un unique 1 sur une colonne de taille M. Par conséquent, si on se donne un problème P, tout algorithme A_{eval} donne naissance à un algorithme A_{test} où chaque appel à f dans A_{eval} est remplacé par $\sqrt{M}$ requêtes dans A_{test}. Si on note $C_{\text{eval}}(P)$ (resp. $C_{\text{test}}(P)$) la complexité de P dans le modèle d'évaluation (resp. de test), on a alors :

Proposition 3.3 *Pour tout problème $P : [1, M]^{[1,N]} \to [1, n]$:*

$$C_{eval}(P) \leq C_{test}(P) \leq \sqrt{M} C_{eval}(P)$$

On aimerait avoir des relations plus précises que ces trivialités ; malheureusement, nous allons voir qu'il existe des problèmes P et P' tels que :

$$C_{\text{test}}(P) = C_{\text{eval}}(P) \quad \text{et} \quad C_{\text{test}}(P') = \sqrt{M} C_{\text{eval}}(P')$$

Comme exemple de problème dont la complexité est identique dans les deux modèles, on considère $P(f) = 1 \Leftrightarrow f(1) = 1$. Il est évident que $C_{\text{test}}(P) = C_{\text{eval}}(P) = 1$, car il suffit d'évaluer $\delta_{1,1}(f)$ pour connaître la valeur de $P(f)$. Les problèmes que nous étudierons par la suite seront des exemples de séparations possibles (en particulier $\sqrt{M}$) entre les deux modèles. Comme il n'existe manifestement pas de relation directe plus précise entre C_{eval} et C_{test}, une idée serait de trouver une mesure de sensitivité d'un problème permettant de relier ces deux complexités. Malheureusement, nous n'avons aucun résultat dans cette direction jusqu'à présent.

3.1 Un exemple : l'égalité de deux fonctions

3.1.1 Présentation du problème

Le premier problème que nous allons étudier est le suivant : on considère deux fonctions $f_1, f_2 : [1, N] \to [1, M]$, et l'on veut savoir si $f_1 = f_2$. On note ce problème EQ, et l'on a donc :

$$EQ(f_1, f_2) = 1 \Longleftrightarrow f_1 = f_2$$

Notons que l'on reste bien dans le cadre défini précédemment ; bien que l'on considère 2 fonctions f_1 et f_2, on peut formellement les grouper en une seule définie sur un espace 2 fois plus grand, fonction que l'on notera alors $f_1 f_2$[18]. Cet exemple très simple va nous permettre de nous familiariser avec le modèle de test.

3.1.2 Résolution dans le modèle d'évaluation

Nous allons montrer ici que $C_{\text{eval}}(EQ) = \Theta(\sqrt{N})$.

Algorithme La borne supérieure est une simple application de l'algorithme de Grover : on cherche i tel que $f_1(i) \neq f_2(i)$. Comme l'évaluation de f_1 et f_2 se fait en temps constant, cet algorithme a donc une complexité de $O(\sqrt{N})$.

Borne inférieure On utilise la méthode d'Ambainis, plus particulièrement le théorème 2.4 avec

$$X = \{f_1 f_2 \mid f_1 = f_2\}$$

$$Y = \{g_1 g_2 \mid g_1 = g_2 \text{ sauf en exactement 1 point.}\}$$

$$R_X(f_1 f_2) = \text{"On change exactement 1 valeur de } f_1 f_2\text{."}$$

$$R_Y(g_1 g_2) = \text{"On change } g_1 \text{ ou } g_2 \text{ au point où ces deux fonctions diffèrent."}$$

Il est immédiat que $1 = EQ(X) \neq EQ(Y) = 0$; c'est une des conditions nécessaires à l'application du théorème. Par la suite, comme elle sera dans tous les cas tout aussi évidente à vérifier, on la considérera comme implicite.

Avec cette définition, on a alors $m = 2N(M-1)$, $m' = 2$, $l = M-1$ et $l' = 1$, ce qui nous donne une borne inférieure de $\Omega\left(\sqrt{\frac{4N(M-1}{M-1}}\right) = \Omega(\sqrt{N})$, ce qui est bien le résultat recherché.

3.1.3 Résolution dans le modèle de test

Nous allons maintenant montrer que le modèle de test est bien moins efficace pour ce problème ; en effet, on a $C_{\text{test}}(EQ) = \Theta(\sqrt{NM})$.

Borne supérieure Elle est immédiate ; en effet, d'après la proposition 3.3 :

$$C_{\text{test}}(EQ) \leq \sqrt{M} C_{\text{eval}}(EQ) = \sqrt{NM}$$

Borne inférieure On va avoir besoin pour la première fois de la deuxième version de la méthode d'Ambainis, c'est-à-dire que l'on va utiliser le théorème 2.5. Pour ce faire, on définit :

$$X = \{f_1 f_2 \mid f_1 = f_2\}$$

$$R_X(f_1 f_2) = \{\text{On choisit un point où l'on change } f_1 \text{ et } f_2 \text{ en des valeurs différentes.}\}$$

On a alors :

$$Y = \{g_1 g_2 \mid g_1 \text{ et } g_2 \text{ diffèrent en exactement 1 point.}\}$$

$$R_Y(g_1 g_2) = \{\text{On change } g_1 \text{ et } g_2 \text{ au point où elles diffèrent vers une valeur commune.}\}$$

[18] Et si l'on se donne U_{f_1} et U_{f_2}, on peut en déduire $U_{f_1 f_2}$ en ajoutant un qubit de contrôle ; cela implique bien entendu de savoir réaliser les transformations $CONTROLLED - U_{f_1}$ et $CONTROLLED - U_{f_2}$, ce qui est toujours possible si U_{f_1} et U_{f_2} sont donnés sous la forme de circuits.

La première chose que l'on remarque sur ces définitions est la différence avec le modèle d'évaluation. En effet, on se permet ici de changer les valeurs des deux fonctions au même point, au lieu d'une seule. La raison d'un tel changement est de permettre plus de possibilités de passer de X à Y et vice-versa. En effet, nous allons voir que le modèle de test génère plus de contraintes que le modèle d'évaluation lors du calcul de l, l' ou $l_{\max}$; si ces valeurs sont trop petites (par exemple constantes comme le l' de la borne inférieure exhibée pour ce problème dans le modèle d'évaluation), ces contraintes supplémentaires n'auront aucun effet sur le calcul de la borne inférieure. En particulier, on peut vérifier qu'utiliser les mêmes paramètres que dans le modèle d'évaluation permet d'obtenir la même borne inférieure, à savoir $\Omega(\sqrt{N})$, et ce que l'on utilise le théorème 2.4 ou 2.5 ; ce n'est donc pas suffisant.

Par ailleurs, notons que formellement, on définit en fait ces ensembles et relations sur les matrices $\delta_{j,i}$; par exemple, changer une valeur de j_1 à j_2 en un point revient à faire l'opération suivante d'échange sur la colonne i :

$$\begin{array}{c} 1 \\ \vdots \\ j_1 \\ \vdots \\ j_2 \\ \vdots \\ M \end{array} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

m et m' sont toujours aussi faciles à calculer : $m = 2N(M-1)(M-2)$ et $m' = (M-2)$.

C'est en essayant de calculer l et l' que l'on se rend compte que le fait d'être dans le modèle de test va jouer un rôle fondamental. En effet, comme expliqué précédemment, un oracle de test est équivalent à un oracle d'évaluation sur une matrice booléenne ayant des contraintes fortes (plus précisément ayant un et un seul 1 par colonne). Si l'on note (j, i) le point en lequel on impose un changement, on a deux cas à examiner :

1. soit $\delta_{j,i} = 1$, que l'on force donc à changer en 0, auquel cas on impose seulement à la fonction correspondante à δ de changer de valeur en i ; il n'y a donc pas de gain par rapport au modèle d'évaluation ;
2. soit $\delta_{j,i} = 0$, que l'on force à changer en 1 ; cela oblige la fonction à changer sa valeur au point i par j, opération que l'on notera $f(i) \leftarrow j$. Il y a donc une contrainte supplémentaire, et beaucoup moins de possibilités de changement ; et comme l et l' sont au dénominateur de la borne inférieur, cela correspond en fait à un gain par rapport au modèle d'évaluation.

Ainsi, il semble qu'en utilisant seulement la version allégée de la méthode d'Ambainis, on puisse obtenir une amélioration de la borne inférieure ; cependant, il y a un piège, car l (resp. l') est définie comme le nombre *maximum* de choix possibles pour une certaine f (resp. g) d'évoluer en g (resp. f) en forçant une certaine valeur à changer. Par conséquent, on est obligé de considérer le pire cas, à savoir celui où $\delta_{j,i} = 1$, et on ne gagne rien.

L'idée est alors d'utiliser la version complète de la méthode d'Ambainis, c'est-à-dire le théorème 2.5. En effet, dans cette version, on se restreint aux fonctions f et g en relation, et aux points où f et g diffèrent ; soit (j, i) l'un de ces points. Comme $\delta_{j,i}$ ne peut prendre que deux valeurs, 0 ou 1, l'une des deux matrices correspondantes à f et g aura nécessairement la valeur 0 en ce point. Ceci nous permet donc de bénéficier en partie du bonus dû au modèle de test - autrement dit, il est possible, lorsque l'on considère un point i où f et g diffèrent, de *forcer* l'une de ces deux fonctions à prendre la valeur de l'autre en ce point.

Par exemple, si $\delta_{j,i}(f) = 1$ et $\delta_{j,i}(g) = 0$, forcer les matrices correspondantes à f ou g à changer en (j, i) revient à changer $f(i)$ sans autre contrainte supplémentaire, et à changer $g(i)$ en $j = f(i)$, ce que l'on notera $g(i) \leftarrow f(i)$. On aura donc à chaque fois deux cas à examiner, suivant que $f(i) \leftarrow g(i)$ ou $g(i) \leftarrow f(i)$; comprendre cet effet créé par le modèle de test est absolument essentiel pour la suite.

Pour calculer $l_{\max}$, on considère donc $f_1 f_2 \in X$, $g_1 g_2 \in Y$, tels que $(f_1 f_2, g_1 g_2) \in R$, c'est-à-dire qu'il existe un index i_0 tel que $f_1(i_0) \neq g_1(i_0)$ et $f_2(i_0) \neq g_2(i_0)$. Par conséquent les $\delta(f_1 f_2)$ et $\delta(g_1 g_2)$ diffèrent exactement en quatre points : $(f_1(i_0), i_0)$, $(g_1(i_0), i_0)$, $(f_2(i_0), i_0)$ et $(g_2(i_0), i_0)$. Sans perte de généralité, on se restreint à l'étude des deux premiers cas (car f_1 et f_2 - resp. g_1 et g_2 - jouent des rôles symétriques) :

1. dans ce cas, on a : $\delta_{f_1(i_0),i_0}(f_1 f_2) = 1$ et $\delta_{f_1(i_0),i_0}(g_1 g_2) = 0$; cela correspond au cas $g_1(i_0) \leftarrow f_1(i_0)$. On a alors de manière immédiate $l = (M-1)(M-2)$ (on choisit vers quelle valeur changer f_1 et f_2 mais pas où l'on change cette valeur) et $l' = 1$ (le changement de g_1 est complètement déterminé - en i_0 vers $f(i_0)$ -, et g_2 doit suivre pour aboutir à deux fonctions égales) ;
2. dans ce cas, on a $\delta_{g_1(i_0),i_0}(f_1 f_2) = 0$ et $\delta_{g_1(i_0),i_0}(g_1 g_2) = 1$, ce qui correspond au cas $f_1(i_0) \leftarrow g_1(i_0)$. On calcule alors de même $l = M - 2$ (on choisit seulement la nouvelle valeur de f_2 en i_0, puisque f_1 doit prendre la valeur $g_1(i_0)$ en i_0) et $l' = M - 2$ (on choisit la nouvelle valeur commune de g_1 et g_2 au point i_0).

On obtient ainsi finalement $l_{\max} = (M-1)(M-2)$ et donc une borne inférieure en

$$\Omega\left(\sqrt{\frac{NM^3}{M^2}}\right) = \Omega(\sqrt{NM})$$

3.1.4 Conclusion

Cette borne inférieure correspond à la borne supérieure présentée précédemment ; par conséquent :

$$C_{\text{test}}(EQ) = \Theta(\sqrt{NM})$$

On a ainsi un exemple de problème plus difficile dans le modèle de test que dans le modèle d'évaluation, et où la différence de complexité est maximale, c'est-à-dire $\sqrt{M}$.

3.1.5 Parenthèse : parallélisation de la méthode d'Ambainis

Cette partie a été réalisée en collaboration avec Yves Verhoeven.

Le meilleur algorithme pour déterminer l'égalité de deux fonctions $[1, N] \to [1, M]$ dans le modèle de test consiste en fait à rechercher un point, c'est-à-dire une colonne dans les matrices associées, où les deux fonctions diffèrent, et ce avec l'algorithme de Grover en $\Theta(\sqrt{N})$; vérifier que les fonctions diffèrent en un point revient à rechercher un indice de ligne dans les colonnes correspondantes où ces dernières diffèrent, ce qui se fait aussi de manière optimale avec l'algorithme de Grover en $\Theta(\sqrt{M})$. On a ainsi l'impression de résoudre deux problèmes indépendants ; en quelque sorte on parallélise le deuxième problème (sur une seule colonne) en N instances, que l'on combine ensuite avec l'algorithme de Grover. On aimerait donc pouvoir automatiser ce processus ; et il se trouve que c'est possible, mais à certaines conditions assez fortes.

On suppose donc avoir un problème $P : [1, K]^{[1,M]} \to \{0, 1\}$, sur lequel on a déjà montré une borne inférieure $B(P)$ par la méthode d'Ambainis, ce qui est assez contraignant. On considère les paramètres X, Y, R, m, m', l et l' de la preuve, avec $P(f) = 0$ pour $f \in X$ et $P(g) = 1$ pour $g \in Y$ (X représente donc des échecs, et Y des succès) ; on a donc $B(P) = \sqrt{\frac{mm'}{ll'}}$. Soit maintenant la parallélisation, notée $Q : \left([1,K]^{[1,M]}\right)^N \to \{0,1\}$, de ce problème en N exemplaires que l'on cherche à résoudre simultanément ; formellement, on a :

$$Q(f_1, \dots, f_N) = P(f_1) \wedge \dots \wedge P(f_N)$$

On pose alors :

$$X_Q = \{(h_1, \dots, h_N) \mid \exists i_0 \in [1,N],\ (h_{i_0} \in X \ \text{ et } \ \forall i \neq i_0,\ h_i \in Y)\}$$

$$Y_Q = Y^N$$

$$((h_1, \dots, h_N), (h'_1, \dots, h'_N)) \in R_Q \iff \forall i \left\{ \begin{array}{l} h_i \in X \Rightarrow (h_i, h'_i) \in R \\ h_i \in Y \Rightarrow h_i = h'_i \end{array} \right\}$$

X_Q est un ensemble défini à partir de X et Y où exactement un seul des sous-problèmes P a échoué ; par conséquent tout élément $f_Q \in X_Q$ vérifie $Q(f_Q) = 0$. A l'opposé, les éléments de y_Q n'ont aucun échec, et donc pour tout $g_Q \in Y_Q$, $Q(g_Q) = 1$. On a donc bien vérifié la première condition du théorème 2.4. On considère maintenant $f_Q \in X_Q$, i_0 l'indice du sous-problème de f_g où il y a un échec, et $g_Q \in Y_Q$. f_Q et g_Q sont alors en relation d'après R_Q si f_Q est égal à g_Q sauf à l'endroit où f_Q échoue, auquel cas on se réfère à R pour déterminer quel type de relation est autorisé. Les coefficients se calculent alors immédiatement :

$$m_Q = Nm \quad m'_Q = m' \quad l_Q = l \quad l'_Q = l'$$

ce qui nous donne donc bien une borne inférieure en $\Omega(\sqrt{N}B(P))$.

Il est probablement possible d'étendre ce résultat (notamment en considérant non plus seulement la fonction $AND = \wedge$, mais aussi n'importe quelle fonction ϕ, pour combiner les n instances de P), mais l'énoncé en deviendrait alors beaucoup trop compliqué pour pouvoir être utile.

3.2 Problème des collisions

On examine maintenant un problème plus intéressant, qui a en particulier des applications en cryptographie, notamment pour évaluer la difficulté d'inverser des fonctions de hachage.

3.2.1 Présentation du problème

On se donne une fonction $f : [1,N] \to [1,M]$, avec N pair et $M \geq N$. On dit que f est 2-to-1 si et seulement si chaque élément de $[1,M]$ a exactement soit 0, soit 2 antécédents par f. On suppose que f est soit injective (on dira aussi 1-to-1), soit 2-to-1 ; le problème des collisions, noté COL consiste alors à déterminer dans lequel de ces deux cas on se trouve.

3.2.2 Résolution dans le modèle d'évaluation

Nous allons voir ici que $C_{\text{eval}}(COL) = \Theta(N^{1/3})$.

Algorithme Pour la borne supérieure, on considère l'algorithme suivant qui recherche une collision dans une fonction f 2-to-1, et paramétré par un entier k :

1. On calcule f en k valeurs quelconques, et on stocke les résultats. Cela se fait en $O(k)$.
2. On cherche i dans $[1,N]$ tel que $f(i) \in f([1,k])$; comme f est 2-to-1, il y a k solutions, par conséquent, avec l'algorithme de Grover, cela se fait en $O(\sqrt{N/k})$.

Remarquons que cet algorithme peut être aisément utilisé pour déterminer si une fonction donnée est 1-to-1 ou 2-to-1 ; il suffit de l'appliquer en supposant que la fonction est 2-to-1, et de vérifier si la solution renvoyée est bien une collision. On obtient donc une complexité globale de $O(k+\sqrt{N/k})$. Pour optimiser ce résultat, on pose $k = N^{1/3}$, ce qui nous donne alors une complexité en $O(N^{1/3})$.

Borne inférieure La borne inférieure est beaucoup plus difficile à réaliser, et utilise la méthode des polynômes. On considère donc un algorithme quantique $\mathcal{A}$ résolvant ce problème en T requêtes ; on sait alors que la probabilité de succès est un polynôme en les $\delta_{j,i}(f)$ de degré au plus $2T$, que nous noterons $P(f)$. Cette présentation est basée sur la preuve de Kutin[12].

L'étape suivante, et probablement la plus compliquée, consiste à réduire le nombre de variables le plus possible ; en effet, le résultat d'approximation que nous possédons, le théorème de Paturi (2.8), ne s'applique qu'aux polynômes à une seule variable. Cependant, comme nous le verrons plus tard, une utilisation astucieuse de ce théorème permet la plupart du temps de se permettre des polynômes à deux ou trois variables.

La méthode la plus pratique pour diminuer le nombre de variables est de *symétriser* le polynôme ; c'est-à-dire que l'on va s'arranger pour faire jouer le même rôle à toutes les entrées et toutes les sorties. En effet, la propriété que l'on cherche à établir, à savoir être 1-to-1 ou 2-to-1, n'a aucune raison de dépendre spécifiquement d'une valeur d'entrée ou de sortie donnée. En particulier, si l'on échange les valeurs de f en deux points i et j, le résultat de l'algorithme devrait rester inchangé.

C'est pourquoi on définit, pour tout $\sigma \in \Sigma_N$ (le groupe de permutation de $[1, N]$) et $\tau \in \Sigma_M$ l'opération :

$$\Gamma_\tau^\sigma(f) = \tau \circ f \circ \sigma$$

Cette opération modifie un oracle f en permutant son entrée et sa sortie. Le principe est ensuite de considérer l'espérance du polynôme P lorsque σ et τ parcourent leurs espaces respectifs, ce que l'on appellera la *symétrisation* de P :

$$\tilde{P}(f) = \mathbb{E}_{\sigma,\tau}(P(\Gamma_\tau^\sigma(f)))$$

Comme toutes les valeurs d'entrée et de sortie peuvent être permutées n'importe comment, cette quantité ne dépend que de la structure de f, à savoir le nombre d'antécédents qu'a chaque valeur de sortie. L'astuce maintenant est de caractériser cette structure par un nombre limité de variables, et de telle sorte que le résultat obtenu soit un polynôme en les-dites variables ; notons qu'il n'est pas nécessaire de considérer tous les oracles possibles et qu'il est même préférable de se limiter à certaines fonctions respectant intuitivement la structure du problème étudié, ici COL.

Dans cette optique, on peut penser ne considérer que les oracles 1-to-1 ou 2-to-1 ; mais dans ce cas, cela ne nous donne que deux valeurs possibles de $\tilde{P}(f)$; ce n'est donc pas suffisant, puisqu'il nous faut au moins une variable. Le seul facteur ayant été modifié ici est le passage de *1*-to-1 à *2*-to-1. On pense donc naturellement à considérer de manière générale les fonctions k-to-1, c'est-à-dire telles que chaque élément de $[1, M]$ a exactement 0 ou k antécédents. Cependant, cela pose un problème majeur : k ne divise pas forcément N, et donc il peut ne peut pas exister de fonction k-to-1 définie sur $[1, N]$; pour résoudre ce problème, Shi[15] propose de considérer en général des fonctions partiellement k-to-1, et injective sur la partie restante. Mais ce n'est pas suffisant pour obtenir la borne inférieure que nous recherchons, à savoir $\Omega(N^{1/3})$.

C'est pourquoi on va encore élargir un peu la classe de fonction considérée, et au lieu de se limiter à des fonctions k-to-1 sur une partie de leur entrée et 1-to-1 sur le reste, on considère des fonctions a-to-1 sur une partie de leur entrée et b-to-1 sur le reste. On remarque que cette définition a l'avantage d'être symétrique. Formellement, on dit qu'un triplet d'entiers (m, a, b) est valide si :

$$\begin{gathered} m \in [1, n] \\ a \text{ divise m} \\ b \text{ divise } N - m \end{gathered}$$

Pour un tel triplet, on définit la fonction :

$$f_{m,a,b} = \begin{cases} \lceil i/a \rceil & \text{pour } 1 \leq i \leq m \\ M - \lfloor (N-i)/b \rfloor & \text{pour } m < i \leq N \end{cases}$$

$f_{m,a,b}$ est donc a-to-1 sur $[1, m]$ et b-to-1 sur $[N - m, N]$.

On peut alors montrer que $Q(m, a, b) = \mathbb{E}_{\sigma,\tau}(P(\Gamma^{\sigma}_{\tau}(f_{m,a,b})))$ est un polynômes en m, a et b de même degré que P, c'est-à-dire au plus $2T$. Notons d ce degré. Q vérifie alors les propriétés suivantes :

1. $0 \leq Q(m, a, b) \leq 1$ si (m, a, b) est valide ; en effet, $Q(m, a, b)$ est la probabilité de réussite de l'algorithme $\mathcal{A}$ sur une entrée qui ne respecte pas forcément la promesse d'être 1-to-1 ou 2-to-1 ;
2. $0 \leq Q(m, 1, 1) \leq 1/3$ pour tout m ; c'est le cas des fonctions 1-to-1.
3. $2/3 \leq Q(m, 2, 2) \leq 1$ pour tout m pair ; c'est le cas des fonctions 2-to-1.

Remarquons que Q est un polynôme, défini pour n'importe quelles valeurs d'entrée, en particulier lorsque ces dernières ne forment pas un triplet valide, ou même lorsqu'elles sont réelles ; cependant, on n'a aucune information sur le comportement de Q en-dehors des cas ci-dessus.

On pose maintenant $m_0 = 2\lfloor n/4 \rfloor$, qui est proche de la moitié de n tout en restant pair. Pourquoi proche du milieu ? C'est pour que le théorème d'approximation 2.8 soit le plus efficace possible. On considère maintenant $Q(m_0, 1, 2)$, et on distingue deux cas :

1. $Q(m_0, 1, 2) \geq 1/2$.

   Alors on considère c le plus petit entier tel que

   $$|Q(m_0, 1, c)| \geq 2$$

   Cet entier existe car $Q(m_0, 1, \_)$ est un polynôme non constant, et par conséquent non borné. Alors par définition, pour x entier dans $[1, c[$ on a :

   $$|Q(m_0, 1, x)| \leq 2$$

   Par ailleurs :

   $$|Q(m_0, 1, 2) - Q(m_0, 1, 1)| \geq 1/6$$

   car par hypothèse $Q(m_0, 1, 2) \geq 1/2$. Par conséquent, $Q(m_0, 1, \_)$ est borné sur un intervalle de longueur c et possède un saut près d'une des extrémités de cet intervalle ; d'après le théorème 2.8, $d = \Omega(\sqrt{c})$.

   Par ailleurs, si l'on considère à présent le polynôme $h(i) = (ci, 1, c)$ alors :

   $$0 \leq h(i) \leq 1 \;\text{ pour tout entier }\; i \in [0, \lfloor N/c \rfloor]$$

   $$|h(m_0/c)| = |Q(m_0, 1, c)| \geq 2$$

   Par conséquent, h est borné sur un intervalle de longueur N/c et possède un saut près du milieu de cet intervalle ; par le même théorème 2.8, $d = \Omega(N/c)$.

   En combinant ces deux résultats, on obtient $d^3 = \Omega(N)$ et donc $d = \Omega(N^{1/3})$.

2. $Q(m_0, 1, 2) \leq 1/2$.

   La borne inférieure se démontre de manière presqu'identique au premier cas en considérant c le plus petit entier tel que $|Q(m_0, c, 2)| \geq 2$ et $h(i) = Q(ci, c, 2)$.

Ceci conclut la preuve de la borne inférieure ; cet exemple montre que la méthode des polynômes n'est pas toujours facile à appliquer, et réduit en fait la recherche d'une borne inférieure à des problèmes d'approximation de polynômes.

3.2.3 Résolution dans le modèle de test

Nous allons montrer ici que $C_{\text{test}}(COL) = \Theta(\sqrt{M})$.

Algorithme L'algorithme est le même que dans le modèle d'évaluation, mais nous choisissons un paramètre k différent :

1. On calcule f en k valeurs quelconques, et on stocke les résultats. Cela se fait en $O(k\sqrt{M})$. Le calcul de la valeur de f en un point est plus coûteux qu'auparavant.
2. On cherche i dans $\{1, \ldots, N\}$ tel que $f(i) \in f([1, k])$; f est 2-to-1, donc il y a k solutions ; il n'est pas besoin ici de calculer à chaque fois la valeur de $f(i)$, il suffit d'effectuer la recherche sur les lignes correspondant aux éléments de $f([1, k])$. Par conséquent, cela se fait en $O(\sqrt{N})$: on cherche un 1 dans une matrice $k \times N$, sachant qu'il y a k solutions.

On obtient donc une complexité globale de $O(k\sqrt{M} + \sqrt{N})$. Comme $M \geq N$, c'est minimal pour $k = 1$, ce qui donne une complexité en $O(\sqrt{M})$.

Borne inférieure Pour la borne inférieure, on utilise de nouveau la méthode d'Ambainis, qui semble marcher très bien pour le modèle de test. Cependant, nous allons devoir examiner deux différents cas ici, suivant que $N \leq M \leq 3N/2$ ou $M > 3N/2$. Il existe probablement un choix de X, Y et R permettant de résoudre conjointement ces deux cas, mais les problèmes de dénombrement posés par le calcul de m, m' et $l_{\max}$ sont assez dissuasifs, surtout dans le cas du problème de collisions.

On pose $p = N/2$.

1. $M \leq 3N/2$. On pose alors :

$$X = \{f \mid f \text{ est 1-to-1.}\}$$
$$Y = \{g \mid g \text{ est 2-to-1.}\}$$
$$R_X(f) = \text{"On change } p = N/2 \text{ valeurs de } f \text{ pour arriver à une fonction 2-to-1."}$$

$$R_X \ : \quad f = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} = g$$

   On a alors :

$$m = \frac{N!}{p!}$$
$$m' = 2^p \frac{(M-p)!}{(M-N)!}$$

   On considère $f \in X$, $g \in Y$ tels que $(f, g) \in R$, et i tel que $f(i) \neq g(i)$. Alors on a deux cas à examiner :

(a) On force f à prendre la valeur $g(i)$ en i : $f(i) \leftarrow g(i)$. Dans ce cas, on a :

$$l_{f,i} = \frac{(N-2)!}{(p-1)!}$$

$$l'_{g,i} = 2^{p-1}\frac{(M-p)!}{(M-N)!}$$

(b) $g(i) \leftarrow f(i)$. Alors :

$$l_{f,i} = \frac{(N-1)!}{(p-1)!}$$

$$l'_{g,i} = 2^{p-1}\frac{(M-p-1)!}{(M-N)!}$$

Cela nous donne finalement :

$$l_{\max} = 2^{p-1}\frac{(N-2)!(M-p-1)!}{(p-1)!(M-N)!}\max(N-1, M-p)$$

$N \leq M \leq 3N/2$ et $p = N/2$, donc $M-p = \Theta(N)$ et $M = \Theta(N)$, et on obtient donc la borne inférieure voulue, à savoir $\Omega(\sqrt{N}) = \Omega(\sqrt{M})$. On remarque ici que le choix naturel (c'est-à-dire le plus simple) de la relation R suffit à montrer la borne inférieure pour M pas trop grand par rapport à N. Pour M plus grand, on est donc obliger de considérer une autre relation.

2. $M > 3/2N$. Dans ce cas, on choisit une différente relation R :
$R_X(f)$ = "On change toutes les valeurs de f vers une fonction g 2-to-1 de telle sorte que $f([1,N])$ et $g([1,N])$ soient disjoints."

$$R_X \ : \quad f = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = g$$

La condition sur $f([1,N])$ et $g([1,N])$ vise à rendre le calcul des paramètres du théorème 2.5 plus facile ; on remarque que cette condition ne peut être vérifiée qu'à condition que $M > 3N/2$:

$$m = \frac{N!}{2^p p!}\frac{(M-N)!}{(M-N-p)!}$$

$$m' = \frac{(M-p)!}{(M-p-N)!}$$

On considère deux fonctions f et g en relation, et i tel que $f(i) \neq g(i)$. Là encore, on a deux cas :

– $f(i) \leftarrow g(i)$; alors :

$$l_{f,i} = \frac{N!}{2^p p!}\frac{(M-N-1)!}{(M-N-p)!}$$

$$l'_{g,i} = \frac{(M-p)!}{(M-p-N)!}$$

– $g(i) \leftarrow f(i)$:

$$l_{f,i} = \frac{N!}{2^p p!} \frac{(M-N)!}{(M-N-p)!}$$

$$l'_{g,i} = \frac{(M-p-1)!}{(M-p-N)!}$$

Cela nous donne :

$$l_{\max} = \frac{N!(M-N-1)!(M-p)!}{2^p p!((M-N-p)!)^2}$$

Et la borne inférieure $\Omega(\sqrt{M-N})$. Mais $M > 3N/2$, donc $M - N > M/3$, ce qui nous permet d'obtenir finalement la borne inférieure en $\Omega(\sqrt{M})$ recherchée.

3.2.4 Conclusion

Le problème des collisions fournit donc un exemple de problème où la séparation entre le modèle de test et le modèle d'évaluation n'est pas tendue. On a en effet, dans le cas où $M = N$:

$$C_{\text{test}}(COL) = N^{1/6} C_{\text{eval}}(COL)$$

Dans le cas général, la perte due au passage dans le modèle de test est de $\frac{\sqrt{M}}{N^{1/3}}$.

3.3 Element Distinctness (ED)

3.3.1 Présentation du problème

Le problème d'Element Distinctness est assez proche de celui des collisions. On considère une fonction $f : \{1, \ldots, N\} \to \{1, \ldots, M\}$, avec $M \geq N$. On cherche à savoir si f admet une collision, c'est-à-dire deux entrées i et j distinctes sur lesquelles f prend la même valeur (donc pour lesquelles $f(i) = f(j)$). On notera ce problème ED.

3.3.2 Résolution dans le modèle d'évaluation

Algorithme [7]
L'algorithme pour ED est là aussi paramétré par un entier k. De même que précédemment, l'algorithme présenté ici trouve une collision dans l'hypothèse où f en a au moins une ; il est facile d'en déduire un algorithme de décision de même complexité.

1. On commence par décomposer $[1, N]$ en une partition en N/k sous-ensembles disjoints $U_1, U_2, \ldots, U_{N/k}$ de cardinal k (on considère que k divise N pour simplifier la présentation, mais cela n'a pas d'importance). Cela ne demande aucun appel à f.
2. On cherche maintenant U_i tel qu'il existe un élément de U_i prenant part à une collision ; grâce à l'algorithme de Grover, cela se fait en $O(\sqrt{N/k})$ requêtes (on fait une recherche parmi N/k différents U_i). Pour déterminer si un élément de U_i fait partie d'une collision de f, on procède en deux étapes :
   (a) On calcule f sur toutes les valeurs de U_i, en $O(k)$ appels à f, car $|U_i| = k$.
   (b) On cherche $x \in [1, N]$ tel que $x \in f(U_i)$; en utilisant une fois de plus l'algorithme de Grover, cette recherche s'effectue en $O(\sqrt{N})$ requêtes.

Globalement, on obtient donc une complexité en $O\left(\sqrt{N/k}(k+\sqrt{N})\right)$, qui est minimale pour $k=\sqrt{N}$, pour une complexité finale de l'algorithme en $O(N^{3/4})$. Remarquons que la deuxième étape utilise un algorithme de Grover dont l'oracle, défini par les sous-étapes a et b, n'est pas nécessairement exact ; le théorème 2.2 permet de s'assurer que cela ne pose pas de problème.

Borne inférieure La borne inférieure dans ce cas procède d'une réduction de COL vers ED. En effet, si l'on considère un oracle 2-to-1 $f : [1,N^2] \to [1,M]$, et que l'on recherche une collision de f, il suffit de considérer de manière aléatoire un sous-ensemble de taille N de l'entrée, noté U ; d'après le paradoxe des anniversaires, comme f ne peux prendre que $N^2/2$ valeurs différentes, on trouve une collision dans U (de taille $\sim \sqrt{N^2/2}$) avec probabilité constante. Il suffit donc ensuite de résoudre ED sur U.

Par conséquent, comme $(N^2)^{1/3} = N^{2/3}$ appels à f sont nécessaires pour résoudre COL sur f, au moins autant sont requis pour résoudre ED sur $f_{|U}$. Ceci nous donne donc une borne inférieure en $\Omega(N^{2/3})$. On remarque ici que pour la première fois, la borne inférieure que l'on obtient ne correspond pas à la borne supérieure de l'algorithme décrit précédemment. Cependant, Ambainis a proposé très récemment un algorithme basé sur la notion de marche aléatoire quantique, et résolvant ED en $O(N^{2/3})$, ce qui clôt ce problème, tout au moins lorsque $M \leq N^2$, condition imposée par le passage par COL pour montrer la borne inférieure ; Ambainis a également proposé une variante de la méthode des polynômes pour résoudre le cas particulier où $M < N^2$[1].

On a donc $C_{\text{eval}}(ED) = \Theta(N^{2/3})$.

3.3.3 Résolution dans le modèle de test

Dans le modèle de test, on obtient une complexité en $\Theta(\sqrt{NM})$.

Algorithme Comme dans le cas du problème des collisions, on utilise le même algorithme que pour le modèle d'évaluation, à ceci près que l'on va bien entendu choisir un différent paramètre k.

1. On décompose $[1,N]$ en une partition en N/k sous-ensembles disjoints $U_1, U_2, \ldots, U_{N/k}$ de cardinal k.
2. On cherche U_i tel qu'il existe un élément de U_i prenant part à une collision ; cela se fait en $O(\sqrt{N/k})$ requêtes. Pour déterminer si un élément de U_i fait partie d'une collision de f, on procède en deux étapes :
   (a) On calcule f sur toutes les valeurs de U_i, en $O(k\sqrt{M})$ appels à f.
   (b) On cherche $x \in [1,N]$ tel que $x \in f(U_i)$; de même que dans le cas du problème des collisions, on peut limiter cette recherche aux lignes correspondant aux valeurs de $f(U_i)$. Cette recherche s'effectue donc en $O(\sqrt{kN})$ requêtes.

Cela nous donne donc une complexité en $O\left(\sqrt{N/K}\left(k\sqrt{M}+\sqrt{kN}\right)\right)$, qui est optimale pour $k=1$. On obtient donc une borne supérieure en $O(\sqrt{MN})$.

Borne inférieure Pour la borne inférieure, on utilise de nouveau la méthode d'Ambainis. Afin d'unifier les cas $M=N$ et $M>N$, on utilise les paramètres suivants :

$$X = \{f \mid f \text{ est 1-to-1.}\}$$

$$Y = \{g \mid g \text{ admet exactement une collision.}\}$$

$R =$ "f et g ont exactement deux valeurs distinctes ; et si l'on note (x, y) la collision de g, alors :
- soit $g(x) = g(y) \notin f([1, N])$ (cas A) ;
- soit $g([1, N]) \subset f([1, N])$ (cas B)."

$$R \text{ (cas } A) \;:\quad f = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = g$$

$$R \text{ (cas } B) \;:\quad f = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = g$$

Cette définition de R enlève en fait un hypothétique "cas C", où la collision serait créée dans $f([1, N])$ avec un seul point évoluant à l'intérieur, et où un autre point irait voir à l'extérieur, cas difficile à gérer d'un point de vue combinatoire :

$$R \text{ (cas } C) \;:\quad f = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = g$$

On a alors :

$$\begin{aligned} m &= \frac{N(N-1)}{2}(M-N) \textit{ (cas A) } + N(N-1)(N-2) \textit{ (cas B)} \\ &= N(N-1)(\frac{M+N}{2} - 2) = \Omega(N^2 M) \end{aligned}$$

Pour le calcul de m dans le cas B, remarquons que le seul moyen d'avoir une collision à l'intérieur de $f([1, N])$ en changeant exactement 2 valeurs est de choisir trois entrées x, y et z, et d'effectuer les opérations suivantes : $f(x) \leftarrow f(y)$ et $f(y) \leftarrow f(z)$. Par ailleurs, on a de manière évidente $M + N = \Theta(M)$ (car $M \geq N$). Notons ici que les cas A et B se combinent pour supprimer le terme en $M - N$, qui aurait nécessité d'avoir M grand par rapport à N (plus précisément, $M \geq (1 + \epsilon)N$ pour un certain $\epsilon > 0$).

Par ailleurs :

$$\begin{aligned} m' &= (M-N+1)(M-N) \textit{ (cas A) } + 2(N-2)(M-N+1) \textit{ (cas B)} \\ &= (M-N+1)(M+N-4) = \Omega((M-N+1)M)) \end{aligned}$$

On considère maintenant $f \in X$, $g \in Y$ tels que $(x, y) \in R$ et $i \in [1, N]$ tel que $f(i) \neq g(i)$. Pour le calcul de $l_{\max}$, on a alors deux cas à examiner.

1. f et g sont en relation d'après le cas A, c'est-à-dire que la collision de g se trouve en dehors de $f([1, N])$:

$$f = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = g$$

Alors on a deux sous-cas :

(a) On force $f(i) \leftarrow g(i)$; on a alors :

$$l_{f,i} = (N-1)\ \textit{(cas A)}\ + 0\ \textit{(cas B)}$$

$$\begin{aligned} l'_{g,i} &= (M-N+1)(M-N)\ \textit{(cas A)}\ + (N-2)(M-N+1)\ \textit{(cas B)} \\ &= (M-N+1)(M-2) \end{aligned}$$

(b) On force $g(i) \leftarrow f(i)$. Alors :

$$\begin{aligned} l_{f,i} &= (N-1)(M-N)\ \textit{(cas A)}\ + 2(N-1)(N-2)\ \textit{(cas B)} \\ &= (N-1)(M+N-4) \end{aligned}$$

$$l'_{g,i} = (M-N)\ \textit{(cas A)}\ + 0\ \textit{(cas B)}$$

Cela nous donne un $l_{\max,1} = O(MN(M-N+1))$.

2. f et g sont en relation d'après le cas B, et donc la collision de g se trouve dans $f([1,N])$; dans le schéma suivant, on a mis dans l'ordre les points x, y et z tels que $f(x) \leftarrow f(y)$ et $f(y) \leftarrow f(z)$:

$$\begin{pmatrix} f(x) & f(y) & f(z) \end{pmatrix} \quad : \quad f = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \longrightarrow \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = g$$

Cela nous force donc à multiplier le nombre de sous-cas par deux, suivant que $i = x$ ou $i = y$, car les deux points où f et g diffèrent ne sont plus équivalents dans ces circonstances :

(a) $i = x$ et $f(x) \leftarrow g(x)$. Alors :

$$l_{f,x} = 0\ \textit{(cas A)}\ + 2(N-2)\ \textit{(cas B)}$$

$$l'_{g,x} = 0\ \textit{(cas A)}\ + 2(M-N+1)\ \textit{(cas B)}$$

(b) $i = x$ et $g(x) \leftarrow f(x)$. Alors :

$$\begin{aligned} l_{f,x} &= (N-1)(M-N)\ \textit{(cas A)}\ + 2(N-1)(N-2)\ \textit{(cas B)} \\ &= (N-1)(M+N-4) \end{aligned}$$

$$l'_{g,x} = 0\ \textit{(cas A)}\ + 2\ \textit{(cas B)}$$

(c) $i = y$ et $f(y) \leftarrow g(y)$. Alors :

$$l_{f,y} = l_{f,x} = 2(N-2)$$

$$\begin{aligned} l'_{g,y} &= (M-N+1)(M-N)\ \textit{(cas A)}\ + (N-2)(M-N+1)\ \textit{(cas B)} \\ &= (M-N+1)(M-2) \end{aligned}$$

(d) $i = y$ et $g(y) \leftarrow f(y)$. Alors :

$$l_{f,y} = l_{f,x} = (N-1)(M+N-4)$$

$$l'_{g,y} = 0\ \textit{(cas A)}\ + (M-N+1)\ \textit{(cas B)}$$

Tout ceci nous donne donc finalement : $l_{\max,2} = O(MN(M-N+1))$.

Par conséquent, on a $l_{\max} = O(MN(M-N+1))$, donnant naissance à une borne inférieure en $\Omega(\sqrt{MN})$, ce qui est bien le résultat recherché :

$$C_{\text{test}}(ED) = \Theta(\sqrt{MN})$$

3.3.4 Conclusion

On a ici un nouvel exemple où la séparation entre le modèle d'évaluation et le modèle n'est pas tendue ; dans le cas où $M = N$, elle est de $N^{1/3}$, et dans le cas général de $\frac{\sqrt{M}}{N^{1/6}}$.

3.4 Généralisation d'ED

3.4.1 Présentation du problème

Au lieu de se limiter à rechercher une collision simple, on considère maintenant une k-collision, c'est-à-dire des points $x_1, \ldots, x_k$ tels que $f(x_1) = \ldots = f(x_k)$. Le problème ED_k est alors défini de la manière suivante : on se donne une fonction f avec la promesse que soit f est injective, soit f admet une k-collision, et l'on veut déterminer dans quel cas on se trouve.

3.4.2 Modèle d'évaluation ?

Dans le modèle d'évaluation, le seul résultat connu est assez évident ; à savoir que pour déterminer s'il existe une k-collision, on commence par décomposer l'entrée en $k-1$ sous-ensembles de même taille - d'après le principe des tiroirs, si f admet une k-collision, au moins l'un de ces sous-ensembles de taille $N/(k-1)$ admet une collision simple ; on a donc réduit ED_k à ED. Par conséquent, on a la borne supérieure suivante :

$$C_{\text{eval}}(ED_k) = O\left(\left(\frac{N}{k}\right)^{2/3}\right)$$

Par contre, la meilleure borne inférieure connue est en $\Omega((N/k)^{1/2})$ pour $k \geq 3$, en appliquant par exemple la méthode d'Ambainis ; en effet, la réduction à partir du problème des collisions pour obtenir la meilleure borne inférieure ne fonctionne plus si $k \geq 3$. La complexité de ce problème reste donc mal cernée.

3.4.3 Résolution dans le modèle de test

Dans le modèle de test, on va réussir à résoudre ce problème.

Algorithme On utilise la réduction à ED décrite ci-dessus, ce qui nous donne une complexité en $O\left(\sqrt{\frac{MN}{k}}\right)$.

Borne inférieure Pour la borne inférieure, on utilise une dernière fois la méthode d'Ambainis, en examinant de manière séparée deux cas, comme dans le cas du problème des collisions :

1. Si $M \leq 3N/2$, on utilise les paramètres suivants :

$$X = \{f \text{ n'a pas de collision.}\}$$

$$Y = \{g \text{ a exactement } 1\ k\text{-collision.}\}$$

$R_X(f) =$ "on choisit $k-1$ couples d'entrées (x_i, y_i) tels que pour tout i, $f(y_i) \leftarrow f(x_i)$ et les $f(x_i)$ forment une k-collision avec un certain $f(z)$."

Il s'agit en fait d'une généralisation du cas A de la relation utilisée pour ED, où l'on change non pas 2 valeurs, mais $2(k-1)$. On verra d'ailleurs que le cas B sera similairement généralisé pour $M > 3N/2$, en changeant cette fois k valeurs de f. La possibilité d'unifier ces deux cas

dans le calcul de la borne inférieure de ED, possibilité absente dans la généralisation, vient probablement du fait que 2 est le seul entier tel que $2(k-1) = k$; ainsi, pour la version de base d'ED, on change le même nombre variable dans les deux cas, ce qui fait que les dénombrements qui s'ensuivent sont globalement du même ordre.
Alors le calcul permet d'obtenir les valeurs suivantes :

$$m = \frac{N!}{(N-2k+1)!(k-1)!}$$

$$m' = k\frac{(N-k)!}{(N-2k+1)!}\frac{(M-N+k-1)!}{(M-N)!}$$

$$mm' = \frac{kN!(N-k)!(M-N+k-1)!}{(N-2k+1)!^2(k-1)!(M-N)!}$$

$$l_{\max} = O\left(k\frac{(N-1)!(N-k-1)!(M-N+k-1)!}{(k-1)!(N-2k+1)!^2(M-N)!}\right)$$

Ce qui permet d'obtenir la borne inférieure $\Omega(\sqrt{\frac{N(N-k)}{k}}) = \Omega(\frac{N}{\sqrt{k}})$ pour $k \leq N/2$ (le cas $k \geq N/2$ est trivial), ce qui est correspond bien à la borne supérieur obtenue précédemment, tout au moins pour $N \leq M \leq 3N/2$.

2. Si $M > 3N/2$, alors on change R :

$$R_X(f) = \text{"On change } k \text{ valeurs de } f \text{ vers une collision à l'extérieur de } f([1, N])\text{."}$$

On a alors :

$$m = (M-N)\frac{N!}{k!(N-k)!}$$

$$m' = \frac{(M-N+k-1)!}{(M-N-1)!}$$

$$l_{\max} = \frac{(N-1)!(M-N+k-1)!}{(k-1)!(N-k)!(M-N-1)!}$$

Ce qui donne une bonne inférieure en $\Omega(\sqrt{\frac{(M-N)N}{k}})$=$\Omega(\sqrt{\frac{MN}{k}})$ pour $M > 3N/2$.

3.4.4 Conclusion

On a ainsi exhibé un problème clos dans le modèle de test, mais dont on ne connaît pas encore exactement la complexité dans le modèle d'évaluation.

Conclusion

En conclusion, après avoir défini le modèle de calcul quantique, nous avons présenté l'algorithme de Grover qui est, avec l'algorithme de Shor et plus récemment le calcul adiabatique et les marches aléatoires quantiques, l'une des rares méthodes connues pour concevoir en calcul quantique des algorithmes plus efficaces qu'en calcul classique. Ensuite, nous avons montré les deux principales méthodes d'obtention de bornes inférieures de complexité dans le modèle de la boîte noire : la méthode par adversaire, et la méthode des polynômes. Ceci nous a amené à définir le modèle de test, une variante où l'on modifie le type de requêtes effectuées sur la boîte noire, et qui montre clairement les limites de la méthodes des polynômes pour toute une classe de problèmes.

Une possibilité d'ouverture consiste évidemment à étudier d'autres problèmes dont la complexité, bien que polynomiale, n'est pas encore entièrement cernée, que ce soit dans le modèle de test ou le modèle d'évaluation : on peut donner comme exemple la multiplication de matrices, ou de manière générale des propriétés monotones de graphe, dont on aimerait relier la complexité quantique à des notions de sensitivité ou de certificats. Une autre idée intéressante serait d'étudier d'autres modèles de requêtes, comme par exemple le modèle $BIT-STRING$ qui, sur la requête (i,j), renvoie le $j^{\text{ème}}$ bit de $f(i)$. L'avantage de ce nouveau modèle de requête est qu'il paraît plus susceptible d'être utilisé en pratique que le modèle de test. Tout ceci montre que ce domaine de recherche reste très ouvert, et de plus très actif, car un certain nombre de problèmes encore ouverts au début du stage ont été résolus pendant ce dernier.

Remerciements

Pour l'aide apportée et les discussions partagées au cour du stage :

Frédéric Magniez
Miklos Santha
Yves Verhoeven
Julia Kempe
Et toute l'équipe "Algorithmique et Complexité" du LRI

Références

[1] A. Ambainis. Quantum lower bounds for collision and element distinctness with small range. 2003. quant-ph/0305179.

[2] Andris Ambainis. Quantum lower bounds by quantum arguments. *J. Comput. System Sci.*, 64(4) :750–767, 2002. Special issue on STOC 2000 (Portland, OR).

[3] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Phys. Rev. Ser.*, A52 :3457–3467, 1995. quant-ph/9503016.

[4] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Proceedings of 39th IEEE FOCS*, pages 352–361, 1998. quant-ph/9802049.

[5] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17 :525–532, 1973.

[6] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani Roychowdhury, and Farrokh Vatan. A new universal and fault-tolerant quantum basis. *Inform. Process. Lett.*, 75(3) :101–107, 2000.

[7] H. Buhrman, Ch. Dürr, M. Heilgman, P. Høyer, F. Magniez, M. Santha, and R. de Wolf. Quantum algorithms for element distinctness. *Proceedings of 16th IEEE Conference on Computational Complexity*, pages 131–137, 2001. quant-ph/0007016.

[8] L.K. Grover. A fast quantum mechanical algorithm for database search. *Proceeding of 28th ACM STOC*, pages 212–219, 1996. quant-ph/9605043.

[9] A.S. Holevo. Bounds for the quantity of information transmitted by a quantum communication channel. *Problems of Information Transmission*, 9 :177–183, 1973.

[10] P. Høyer, M. Mosca, and R. de Wolf. Quantum search on bounded-error inputs. 2003. quant-ph/0304052.

[11] Peter Høyer, Jan Neerbek, and Yaoyun Shi. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica*, 34(4) :429–448, 2002. Quantum computation and quantum cryptography.

[12] S. Kutin. A quantum lower bound for the collision problem. 2003. quant-ph/0304162.

[13] Noam Nisan and Márió Szegedy. On the degree of Boolean functions as real polynomials. *Comput. Complexity*, 4(4) :301–313, 1994. Special issue on circuit complexity (Barbados, 1992).

[14] R. Paturi. On the degree of polynomials that approximate symmetric boolean functions. *Proceedings of 24th ACM STOC*, pages 468–474, 1992.

[15] Y. Shi. Quantum lower bounds for the collision and the element distinctness problem. 2001. quant-ph/0112086.

[16] Y. Shi. Both toffoli and controlled-not need little help to do universal quantum computation. 2002. quant-ph/0205115.

[17] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5) :1484–1509, 1997. quant-ph/9508027.

Annexe : démonstration du théorème 2.4

Cette preuve est inspirée de la preuve d'une variante proposée par Høyer[11] et évitant le formalisme de la démonstration d'Ambainis ; nous reprendrons cependant les notations d'Ambainis, plus claires (en évitant en particulier d'introduire inutilement des opérateurs de projection), et adapterons la fin pour retrouver la version originelle du théorème.

On considère l'état des qubits utilisés par le circuit C après k étapes de calcul sur une entrée (classique) h quelconque :

$$|\psi_h^k\rangle = U_k O_h U_{k-1} \dots O_h U_1 O_h U_0 |0\rangle$$

On pose alors :

$$S_k = \sum_{(f,g)\in R} \left|\langle\psi_f^k|\psi_g^k\rangle\right|$$

Remarquons tout de suite que cette expression est invariante sous l'action des portes U_k ; par conséquent, on ne s'intéressera par la suite qu'aux requêtes à l'oracle.

Comme expliqué précédemment, le théorème est conséquence immédiate de ces deux inégalités :

1. $S_0 - S_T \geq (1 - 2\sqrt{\epsilon(1-\epsilon)})\sqrt{|X|\,|Y|\,mm'}$;
2. et $\forall k,\ S_k - S_{k+1} \leq 2\sqrt{|X|\,|Y|\,ll'}$;

dont voici les preuves :

1. Pour montrer cette inégalité, on considère $(f,g) \in R$. Alors :

$$\langle\psi_f^0|\psi_g^0\rangle = 1$$

$$\left|\langle\psi_f^T|\psi_g^T\rangle\right| \leq 2\sqrt{\epsilon(1-\epsilon)}$$

En effet, comme $P(f) \neq P(g)$, on doit pouvoir distinguer $|\psi_f^T\rangle$ et $|\psi_g^T\rangle$ avec probabilité d'erreur au plus ϵ ; il suffit donc d'appliquer le lemme. On a alors

$$S_0 - S_T \geq (1 - 2\sqrt{\epsilon(1-\epsilon)})\,|R|$$

Or, pour tout $f \in X$, il y a au moins m différents $g \in Y$ tels que $(f,g) \in R$; de même avec m' en échangeant f et g. Par conséquent :

$$|R| \geq \max(|X|\,m, |Y|\,m') \geq \frac{|X|\,m + |Y|\,m'}{2} \geq \sqrt{|X|\,|Y|\,mm'}$$

et

$$S_0 - S_T \geq (1 - 2\sqrt{\epsilon(1-\epsilon)})\sqrt{|X|\,|Y|\,mm'}$$

2. On a par définition :

$$\begin{aligned} S_k - S_{k+1} &= \sum_{(f,g)\in R} \left(\left|\langle\psi_f^k|\psi_g^k\rangle\right| - \left|\langle\psi_f^{k+1}|\psi_g^{k+1}\rangle\right| \right) \\ &\leq \sum_{(f,g)\in R} \left(\left|\langle\psi_f^k|\psi_g^k\rangle - \langle\psi_f^{k+1}|\psi_g^{k+1}\rangle\right| \right) \end{aligned}$$

Écrivons $|\psi_f^k\rangle$ dans la base de requêtes :

$$|\psi_f^k\rangle = \sum_{i,z} \alpha_{i,z,f} |i\rangle|z\rangle$$

Alors on a, après la requête (avec S_f) :

$$|\psi_f^{k+1}\rangle = \sum_{i:f(i)=0,z} \alpha_{i,z,f}|i\rangle|z\rangle - \sum_{i:f(i)=1,z} \alpha_{i,z,f}|i\rangle|z\rangle$$

Ceci nous permet à présent de calculer les produits scalaires :

$$\langle\psi_f^k|\psi_g^k\rangle = \sum_{i,z} \overline{\alpha_{i,z,f}}\alpha_{i,z,g}$$

$$\langle\psi_f^{k+1}|\psi_g^{k+1}\rangle = \sum_{i:f(i)=g(i),z} \overline{\alpha_{i,z,f}}\alpha_{i,z,g} - \sum_{i:f(i)\neq g(i),z} \overline{\alpha_{i,z,f}}\alpha_{i,z,g}$$

On obtient donc pour $S_k - S_{k+1}$:

$$\begin{aligned} S_k - S_{k+1} &\leq 2\sum_{(f,g)\in R} \left| \sum_{i:f(i)\neq g(i),z} \overline{\alpha_{i,z,f}}\alpha_{i,z,g} \right| \\ &\leq \sum_{(f,g)\in R} \sum_{i:f(i)\neq g(i),z} \left(\sqrt{\frac{l'}{l}}\left|\alpha_{i,z,f}\right|^2 + \sqrt{\frac{l}{l'}}\left|\alpha_{i,z,g}\right|^2 \right) \\ S_k - S_{k+1} &\leq \sqrt{ll'}\left(\sum_{f\in X\cup Y,i,z} \left|\alpha_{i,z,f}\right|^2 \right) \end{aligned}$$

Mais $\sum_{i,z}\left|\alpha_{i,z,f}\right|^2$ est exactement la norme de $|\psi_f^k\rangle$ au carré, égale à 1 ; on a donc finalement :

$$S_k - S_{k+1} \leq \sqrt{ll'}(|X| + |Y|) \leq 2\sqrt{|X|\,|Y|\,ll'}$$

ce qui conclut la preuve de la deuxième inégalité et du théorème. □