Université de Montréal

Amplification de l'amplitude : analyse et applications

par
Philippe Lamontagne

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Janvier 2013

Résumé

Ce mémoire étudie l'algorithme d'amplification de l'amplitude et ses applications dans le domaine de test de propriété. On utilise l'amplification de l'amplitude pour proposer le plus efficace algorithme quantique à ce jour qui teste la linéarité de fonctions booléennes et on généralise notre nouvel algorithme pour tester si une fonction entre deux groupes abéliens finis est un homomorphisme. Le meilleur algorithme quantique connu qui teste la symétrie de fonctions booléennes est aussi amélioré et l'on utilise ce nouvel algorithme pour tester la quasi-symétrie de fonctions booléennes.

Par la suite, on approfondit l'étude du nombre de requêtes à la boîte noire que fait l'algorithme d'amplification de l'amplitude pour amplitude initiale inconnue. Une description rigoureuse de la variable aléatoire représentant ce nombre est présentée, suivie du résultat précédemment connue de la borne supérieure sur l'espérance. Suivent de nouveaux résultats sur la variance de cette variable. Il est notamment montré que, dans le cas général, la variance est infinie, mais nous montrons aussi que, pour un choix approprié de paramètres, elle devient bornée supérieurement.

Mots clés : Amplification de l'amplitude, test de propriété, test de linéarité, test d'homomorphisme, test de symétrie, test de quasi-symétrie, variance, complexité en requêtes.

Abstract

This thesis studies the quantum amplitude amplification algorithm and some of its applications in the field of property testing. We make use of the amplitude amplification algorithm to design an algorithm testing the linearity of Boolean functions which is more efficient than the previously best known quantum algorithm. We then generalize this new algorithm to test if a function between two finite abelian groups is a homomorphism. We improve on the previously best known algorithm for testing the symmetry of Boolean functions and use this new algorithm to test the quasi-symmetry of Boolean functions.

Next, we further the study of the query complexity of the amplitude amplification algorithm for unknown initial amplitude. We give a rigorous description of the random variable representing the number of queries made by the algorithm and present the previously known result on its expected value upper bound. We then provide new results on the variance of this random variable. It is shown that, in the general case, the variance cannot be bounded above. We show, however, that it can be bounded for an appropriate choice of parameters.

Keywords: Quantum amplitude amplification, property testing, linearity testing, homomorphism testing, symmetry testing, quasi-symmetry testing, variance, query complexity.

Table des matières

Résumé . . . . . **ii**

Abstract . . . . . **iii**

Table des matières . . . . . **iv**

Liste des tableaux . . . . . **vi**

Notation . . . . . **vii**

Chapitre 1 : Introduction **1**

Chapitre 2 : Notions préliminaires **3**

2.1 Le modèle de calcul quantique 3

2.1.1 Qubits et espaces d'Hilbert 3

2.1.2 Transformations unitaires 4

2.1.3 Les mesures 7

2.2 Algorithme d'amplification de l'amplitude 8

Chapitre 3 : Tests de propriétés **12**

3.1 Test de linéarité 12

3.2 Test d'homomorphisme de groupes abéliens 16

3.3 Test de symétrie 19

3.4 Test de quasi-symétrie 22

3.4.1 Estimation de la dépendance 23

3.4.2 L'algorithme 26

Chapitre 4 : Analyse de l'algorithme d'amplification de l'amplitude . . **29**

4.1 La distribution du nombre de requêtes 29

4.2 L'espérance 33

4.3 La variance 34
4.4 Autre travaux 38
4.4.1 Une borne inférieure sur la variance 38
4.4.2 Constante multiplicative 39

Chapitre 5 : Conclusion **41**
5.1 Travaux futurs 42

Bibliographie **43**

Liste des tableaux

5.I Testeurs de propriété décrits dans ce mémoire 41

Notation

$\mathbb{N}$	L'ensemble des entiers positifs $\{1, 2, 3, ...\}$
$\mathbb{C}$	L'ensemble des nombres complexes.
$\|\alpha\|$	La norme du nombre complexe α.
α^*	Conjugué complexe du nombre α.
$\mathcal{H}_2^{\otimes n}$	L'espace de Hilbert à 2^n dimensions.
$[N]$	L'ensemble $\{0, 1, ..., N-1\}$.
$\|A\|$	Cardinalité, ou nombre d'éléments, de l'ensemble A.
$j \in_{\mathrm{R}} A$	j est choisi aléatoirement de manière uniforme parmi les éléments de l'ensemble A.
$\|\| \|\psi\rangle \|\|$	La norme euclidienne du vecteur $\|\psi\rangle$; $\sqrt{\langle\psi \mid \psi\rangle}$
$\mathsf{U}^\dagger$	Transposée conjuguée de U.
$w(x)$	Poids de Hamming de la chaîne booléenne x.
$\mathsf{U}^{\otimes n}$	Le produit tensoriel de n copies de la transformation U.
$\{0,1\}^n$	L'ensemble des chaînes de n bits.

Chapitre 1

Introduction

L'avantage de la mécanique quantique sur la mécanique classique à des fins de calculs ne fait plus aucun doute. L'exemple le plus populaire est l'algorithme de Shor [Sho97] qui permet de factoriser des entiers en temps polynomial par rapport à la taille de ces entiers. Il s'agit là d'une accélération importante par rapport au meilleur algorithme classique connu. Cette découverte vient perturber la cryptographie classique car plusieurs protocoles populaires supposent que le problème de factorisation est difficile. Un ordinateur quantique assez gros compromettrait la sécurité de ces protocoles.

Bien que l'algorithme de Shor témoigne de la puissance de l'ordinateur quantique, il ne montre pas en soit une séparation entre la puissance de calcul des ordinateurs quantiques et classiques. En effet, aucune preuve n'a encore été trouvée qui démontre la difficulté de résoudre le problème de factorisation sur un ordinateur classique et il se pourrait qu'un algorithme efficace existe. L'algorithme le plus connu qui démontre une séparation entre le classique et le quantique est l'algorithme de Grover [Gro97]. Le problème résolu par cet algorithme est de trouver, étant donné une fonction booléenne $f : \{0,1\}^n \to \{0,1\}$, une valeur $x \in \{0,1\}^n$ telle que $f(x) = 1$. Lorsque la fonction f est donnée en boîte noire, c'est-à-dire qu'on ne peut obtenir de l'information sur f qu'en l'évaluant sur ses entrées, il est possible de montrer que tout algorithme classique doit effectuer au moins $\Omega(2^n)$ requêtes à f en moyenne avant de trouver une solution. L'algorithme de Grover requiert $O(2^{n/2})$ requêtes pour accomplir la même tâche.

Ce mémoire étudie une généralisation de l'algorithme de Grover : l'algorithme d'amplification de l'amplitude [BHMT02]. Cet algorithme est présenté à la section 2.2 après une introduction du modèle de calcul quantique qui révisera les concepts essentiels et introduira la notation utilisée dans ce document. Au chapitre 3, l'algorithme d'amplification d'amplitude est utilisé pour améliorer deux algorithmes quantiques résolvant les problèmes de tester la linéarité et la symétrie de fonctions booléennes. Toujours dans ce chapitre, l'algorithme d'amplification de l'amplitude est utilisé pour résoudre deux gé-

néralisations de ces problèmes, soit les problèmes de tester si une fonction sur groupes abéliens finis est un homomorphisme et de tester si une fonction booléenne est quasi-symétrique. Au chapitre 4, on approfondit l'analyse du nombre de requêtes faites par l'algorithme d'amplification de l'amplitude. La variable aléatoire correspondant à ce nombre est d'abord décrite à la section 4.1. La section 4.2 donne une preuve de la borne supérieure sur le nombre espéré de requêtes calculée dans [BBHT98]. Par la suite, dans la section 4.3 de nouveaux résultats sur la variance de cette variable aléatoire sont présentés, notamment, il est démontré que la variance est infinie dans le cas général. Finalement, la conclusion résume les nouveaux résultats qui figurent dans ce mémoire ainsi que des problèmes ouverts découlant de ces résultats.

Chapitre 2

Notions préliminaires

2.1 Le modèle de calcul quantique

Cette section introduit la théorie sous-jacente des résultats présentés dans ce mémoire. Le contenu de cette section ne se veut pas pédagogique, mais sert plutôt à établir les conventions de notation qui seront utilisées tout au long de ce mémoire. Plusieurs livres présentent une description détaillée du modèle de calcul quantique [NC10, Hir04]

2.1.1 Qubits et espaces d'Hilbert

L'élément de base de la théorie de l'information quantique est le *qubit* (pour *quantum bit*). Contrairement à un bit classique qui ne peut prendre que les valeurs 0 et 1, un qubit peut avoir comme état une combinaison linéaire des états classiques $|0\rangle$ et $|1\rangle$. Pour noter un qubit, on utilisera la *notation de Dirac* « $|\cdot\rangle$ ». Dans sa forme la plus générale, un qubit s'écrit donc comme

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

où $\alpha, \beta \in \mathbb{C}$ sont sujet à la restriction $|\alpha|^2 + |\beta|^2 = 1$. Les nombres complexes α et β sont appelés les *amplitudes* de $|0\rangle$ et $|1\rangle$, respectivement. Lorsque α et β sont tous deux non-nuls, on dira que $|\psi\rangle$ est dans une *superposition* de $|0\rangle$ et $|1\rangle$.

Pour soutirer de l'information d'un qubit, on doit le mesurer. De cette opération peut survenir deux résultats ; soit on obtient 0 avec probabilité $|\alpha|^2$, soit on obtient 1 avec probabilité $|\beta|^2$. La restriction imposée à ces deux nombres complexes devient alors nécessaire. Mesurer le qubit le détruit. Toutefois, comme on connaît le résultat de la mesure, on peut créer un qubit dans l'état classique correspondant. On peut donc considérer que la mesure réduit l'état du qubit à l'état classique correspondant au résultat de la mesure, c'est-à-dire qu'obtenir 1 en mesurant $|\psi\rangle$ laisse le qubit dans l'état $|1\rangle$.

On peut aussi avoir des états composés de plusieurs qubits. On considère alors l'état conjoint de deux qubits $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ et $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$ comme étant le *produit tensoriel* des deux qubits $|\psi\rangle \otimes |\phi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$ (par abus de terminologie, on écrira le produit tensoriel de deux états quelconques comme $|\psi\rangle\,|\phi\rangle$).

Le qubit $|\psi\rangle$ peut aussi être perçu comme le vecteur $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ dans l'espace vectoriel $\mathbb{C}^2$. Ainsi, l'état $|0\rangle$ est représenté par le vecteur $\begin{pmatrix}1\\ 0\end{pmatrix}$ et l'état $|1\rangle$ par le vecteur $\begin{pmatrix}0\\ 1\end{pmatrix}$. Le produit tensoriel de deux qubits est alors défini comme

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \otimes \begin{pmatrix}\gamma\\ \delta\end{pmatrix} = \begin{pmatrix}\alpha\gamma\\ \alpha\delta\\ \beta\gamma\\ \beta\delta\end{pmatrix}.$$

Cette définition se généralise de manière intuitive pour les états de dimension arbitraire.

On peut définir un produit interne de deux vecteurs de $\mathbb{C}^2$ par $\langle\psi|\phi\rangle = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}\begin{pmatrix}\gamma\\ \delta\end{pmatrix}$, où $\langle\psi| = |\psi\rangle^\dagger = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix}$ est le vecteur résultant de la transposition et de la conjugaison de $|\psi\rangle$. Lorsqu'on considère $\mathbb{C}^2$ muni de ce produit interne, il devient l'espace d'Hilbert $\mathcal{H}_2$ de dimension 2. On peut augmenter la dimension de cet espace en le multipliant par lui même à l'aide du produit tensoriel. En répétant n fois, on obtient l'espace d'Hilbert $\mathcal{H}_2^{\otimes n} = \mathsf{span}\{|x\rangle \mid x \in \{0,1\}^n\}$ de dimension 2^n qui contient tous les états conjoints possibles de n qubits. La base $\{|x\rangle \mid x \in \{0,1\}^n\}$ est appelée *base de calcul* de $\mathcal{H}_2^{\otimes n}$ dont les éléments coincident avec les chaînes de n bits $x \in \{0,1\}^n$. On peut avoir un espace d'Hilbert de dimension finie quelconque N qu'on écrira alors $\mathcal{H}_N$. Dans ce cas, la base de calcul est $\{|x\rangle \mid x \in [N]\}$ où $[N] = \{0, \ldots, N-1\}$.

2.1.2 Transformations unitaires

Les opérations permises par la mécanique quantique sur des systèmes de qubits sont les transformations dites *unitaires*. Une transformation, représentée sous forme matri-

cielle U, est unitaire si et seulement si son inverse est sa transposée conjuguée ;

$$\mathsf{U}^\dagger \mathsf{U} = \mathbb{I}$$

où $\mathbb{I}$ est la matrice identité.

Le produit tensoriel peut aussi être défini sur les transformations. Le produit tensoriel de deux matrices $\mathsf{U} = \begin{pmatrix} \alpha & \gamma \\ \beta & \delta \end{pmatrix}$ et $\mathsf{V} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ est

$$\mathsf{U} \otimes \mathsf{V} = \begin{pmatrix} \alpha\mathsf{V} & \gamma\mathsf{V} \\ \beta\mathsf{V} & \delta\mathsf{V} \end{pmatrix} = \begin{pmatrix} \alpha a & \alpha c & \gamma a & \gamma c \\ \alpha b & \alpha d & \gamma b & \gamma d \\ \beta a & \beta c & \delta a & \delta c \\ \beta b & \beta d & \delta b & \delta d \end{pmatrix}.$$

Cette définition se généralise de manière intuitive sur les matrices de dimension plus élevée.

Parmi les transformations unitaires notoires, on retrouve la transformation de *Walsh-Hadamard*, $\mathsf{H} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix}$. Il est à noter que la transformation de Walsh-Hadamard est auto-inverse, c'est-à-dire que $H(H\,|\psi\rangle) = |\psi\rangle$ pour tout qubit $|\psi\rangle$. Une des utilités de cette transformation est que, lorsqu'appliquée en parallèle à n qubits dans l'état $|00\ldots0\rangle$ (état que l'on écrira $|0\rangle$ par abus de notation), elle crée une superposition égale de tous les états de la base de calcul :

$$\mathsf{H}^{\otimes n}\,|0\rangle = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle\,.$$

où $\mathsf{H}^{\otimes n} = \mathsf{H} \otimes \mathsf{H} \otimes \ldots \otimes \mathsf{H}$. Lorsque cette *tour* de transformations de Walsh-Hadamard est appliquée à un état de base $|x\rangle$ quelconque, l'état se retrouve dans la superposition

suivante

$$\mathsf{H}^{\otimes n} |x\rangle = \frac{1}{2^{n/2}} \sum_{y} (-1)^{y \cdot x} |y\rangle$$

où $y \cdot x$ est le produit scalaire de deux chaînes booléennes, défini par

$$y \cdot x = \bigoplus_{i=0}^{n-1} y_i \wedge x_i.$$

La tour de transformations de Walsh-Hadamard agit sur un espace d'Hilbert de dimension 2^n. Lorsqu'on travaille dans un espace de dimension N quelconque et qu'on veut obtenir une superposition uniforme de tous les états de la base de calcul, on utilise la transformée de Fourier quantique F_N. La transformée de Fourier quantique est définie, pour les qubits $|x\rangle$ de la base de calcul, par

$$|x\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{y=0}^{N} e^{\frac{2\pi i y x}{N}} |y\rangle$$

où $i = \sqrt{-1}$ et yx est le produit des entiers représentés par les chaînes binaires x et y.

Lorsque l'on souhaite calculer une fonction Booléenne $f : \{0,1\}^n \to \{0,1\}$ quelconque dans ce modèle, on doit utiliser la transformation suivante afin de préserver la condition d'unitarité. La fonction f est implémentée par la transformation O_f qui est défini sur $\mathcal{H}_2^{\otimes n} \otimes \mathcal{H}_2$ par :

$$\mathsf{O}_\mathsf{f} |x\rangle |b\rangle = |x\rangle |b \oplus f(x)\rangle$$

pour $x \in \{0,1\}^n$ et $b \in \{0,1\}$. Lorsque le qubit $|b\rangle$ est dans l'état $\mathsf{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, l'effet de la transformation O_f est

$$\begin{aligned} \mathsf{O}_\mathsf{f}(|x\rangle \otimes \mathsf{H}|1\rangle) &= |x\rangle \otimes \frac{1}{\sqrt{2}}(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) \\ &= (-1)^{f(x)} |x\rangle \otimes |b\rangle . \end{aligned}$$

À la lumière de cette propriété, il est possible de définir l'opérateur d'inversion de la phase conditionel à f noté S_f et défini par $\mathsf{S}_f |x\rangle = (-1)^{f(x)} |x\rangle$ sur les états de base $|x\rangle$.

Au travers de ce mémoire, la mesure de complexité que nous considérerons est la complexité en requêtes ou *query complexity*. Cette mesure de complexité compte le nombre de requêtes qui sont faites à une fonction qui est donnée en boîte noire, ou à un *oracle* comme nous l'appellerons parfois. Les raisons de ce choix sont multiples. D'abord, plusieurs techniques existent pour prouver des bornes inférieures sur la complexité en requêtes (ce qui n'est pas le cas pour la complexité en temps). Ceci permet entre autre de montrer une séparation entre la complexité quantique et classique de certains problèmes. Ces techniques sont aussi utiles pour prouver l'optimalité de certains algorithmes. De plus, les problèmes étudiés au chapitre 3 sont des problèmes de test de propriété. La norme dans ce domaine est d'utiliser le nombre de requêtes effectuées à la fonction boîte noire comme mesure de complexité.

2.1.3 Les mesures

En plus de savoir représenter et manipuler l'information quantique, il faut savoir en extraire de l'information classique soit pour fins de post-traitement ou simplement pour obtenir la sortie d'un algorithme quantique. Il n'est pas possible, par une mesure, de connaître l'état d'un registre quantique, c'est-à-dire, l'amplitude associée à chaque état de base. Si c'était le cas, on pourrait espérer par exemple évaluer une fonction boîte noire $f\colon \{0,1\}^n \to \{0,1\}$ sur une superposition de toutes ses entrées,

$$\mathsf{S}_f \mathsf{H}^{\otimes n} |0\rangle = \frac{1}{2^{n/2}} \sum_x (-1)^{f(x)} |x\rangle ,$$

ce qui nous permettrait de connaître toutes les valeurs que prend f par une seule invocation.

Pour définir rigoureusement une mesure, nous devons définir la notion *d'observable*.

Définition 1. *Un observable $E = \{E_1, \dots, E_m\}$ est une collection de sous-espaces mu-*

tuellement exclusifs (c'est-à-dire que $E_1 \cap ... \cap E_m = \emptyset$) de $\mathcal{H}_N$ tels que

$$\mathcal{H}_N = E_1 \oplus ... \oplus E_m$$

où $m \leq N$ et $\oplus$ est la somme directe *d'espaces vectoriels.*

À chaque sous-espace E_i d'un observable on associe la projection P_{E_i} sur ce sous-espace. On peut représenter un état $|\psi\rangle$ quelconque comme $|\psi\rangle = |\psi_1\rangle + \cdots + |\psi_m\rangle$ où $|\psi_i\rangle \in E_i$. Lorsqu'on mesure $|\psi\rangle$ selon l'observable E, la valeur i est observée avec probabilité $p(i)$ définie par

$$p(i) = || \, |\psi_i\rangle \, ||^2 = \langle \psi \mid \mathsf{P}_{E_i} \mid \psi \rangle = \langle \psi \mid \psi_i \rangle \, .$$

On dira aussi, par abus de terminologie, que l'on observe E_i au lieu de la valeur i qui lui est associée.

On dit que la mesure est *partielle* si elle est effectuée sur un sous-système quantique résidant dans un système plus complexe $|\psi\rangle$, c'est-à-dire, sur un sous-espace stricte $\mathcal{H}' \subset \mathcal{H}_N$. La composante de $|\psi\rangle$ dans $\mathcal{H}'$ est projetée sur l'espace $E_i \subseteq \mathcal{H}'$ correspondant au résultat de la mesure. L'état normalisé après la mesure est

$$\frac{\mathsf{P}_{E_i} |\psi\rangle}{\sqrt{p(i)}}.$$

Lorsque l'on dit que l'on mesure dans une base quelconque $\{|x_1\rangle, ..., |x_n\rangle\}$, cela revient à mesurer l'observable $E = \{E_1, ..., E_n\}$ où $E_i = \mathrm{span}\{|x_i\rangle\}$. À défaut de spécifier un observable, les mesures se feront dans la base de calcul.

2.2 Algorithme d'amplification de l'amplitude

Supposons que l'on dispose d'une fonction boîte noire $f : [N] \to \{0, 1\}$ dont on veut trouver une *solution*, c'est-à-dire une entrée x telle que $f(x) = 1$. Classiquement, on ne peut faire mieux que d'évaluer f sur différents éléments de son domaine. On trouve alors une solution avec un nombre de requête espéré dans l'ordre de N/t si t est le nombre

de solutions. L'algorithme de Grover [Gro97] permet d'améliorer ce nombre à l'ordre de $\sqrt{N/t}$ par l'application répétée d'un opérateur nommé *itération de Grover*.

Supposons que certaines informations soient connues à propos de f, de manière à ce qu'il existe un algorithme quantique ne faisant aucune mesure, représenté par la transformation unitaire A, tel qu'en mesurant l'état $\mathsf{A}\left|0\right\rangle$, on trouve une solution à f avec probabilité $a \geq t/N$. Par une approche classique, on devrait répéter cet algorithme $O(1/a)$ fois en moyenne pour obtenir une solution avec bonne probabilité. L'algorithme d'amplification de l'amplitude [BHMT02] permet de réduire ce nombre à $O(\sqrt{1/a})$. Tout comme Grover, il procède en appliquant successivement l'opérateur défini par

$$\mathsf{G} = -\mathsf{A}\mathsf{S}_0\mathsf{A}^\dagger\mathsf{S}_f$$

où $\mathsf{S}_f\left|x\right\rangle = (-1)^{f(x)}\left|x\right\rangle$, et S_0 multiplie par -1 la phase de l'état $\left|0\right\rangle$ et agît comme l'identité sur les autres états de base. L'effet de cet opérateur sur l'état initial $\left|\psi\right\rangle = \mathsf{A}\left|0\right\rangle$ est d'augmenter l'amplitude des états de base correspondant aux solutions de f. Soit $\left|\psi_1\right\rangle$ la projection de $\left|\psi\right\rangle$ sur le sous-espace engendré par les solutions de f. Soit θ tel que l'amplitude initiale de $\left|\psi_1\right\rangle$ est $\sin(\theta) = \sqrt{a}$. Après j applications de l'opérateur G, cette amplitude devient $\sin((2j+1)\theta)$. Il suffit donc de choisir une valeur de j appropriée dans l'ordre de $1/\theta$ pour que la probabilité de mesurer une solution, $\sin^2((2j+1)\theta)$, soit près de 1.

Puisque la probabilité de succès après j itérations de Grover est une fonction sinusoïdale, un mauvais choix de j entraîne une mauvaise probabilité de succès. Lorsque l'amplitude initiale a est inconnue, il devient impossible de choisir une valeur de j qui donne assurément une bonne probabilité de succès. Cependant, il est possible de contourner cette restriction en choisissant j aléatoirement de manière uniforme entre 0 et un entier $M-1$ (ce qu'on notera $j \in_R [M]$). Lorsque les valeurs possibles de la pige aléatoire sont assez grandes, on peut borner inférieurement la probabilité de mesurer une solution à f. C'est ce qu'ont trouvé Boyer, Brassard, Høyer et Tapp dans [BHMT02] avant même la découverte de l'amplification de l'amplitude. Leur résultat est énoncé dans le lemme suivant en prenant compte du nouveau contexte de l'amplification de l'amplitude.

Lemme 2. *Soit a la probabilité de trouver une solution en mesurant $\mathsf{A}|0\rangle$ et soit $0 < \theta < \pi/2$ tel que $\sin^2\theta = a$. Soit M un entier positif et j un entier choisi au hasard selon la distribution uniforme entre 0 et $M-1$. Après j applications de l'itération de Grover sur l'état initial, la probabilité de mesurer une solution est donnée par*

$$P_M = \frac{1}{2} - \frac{\sin(4M\theta)}{4M\sin(2\theta)}.$$

En particulier, $P_M \geq 1/4$ lorsque $M \geq 1/\sin(2\theta)$.

Démonstration. La probabilité de succès lorsqu'on mesure le registre après $j \in_R [M]$ applications de l'opérateur G sur un état initial $\mathsf{A}|0\rangle$ est

$$\begin{aligned} P_M &= \sum_{j=0}^{M-1} \frac{1}{M}\sin^2((2j+1)\theta) \\ &= \frac{1}{2M}\sum_{j=0}^{M-1} 1 - \cos((2j+1)2\theta) \\ &= \frac{1}{2} - \frac{\sin(4M\theta)}{4M\sin(2\theta)}. \end{aligned}$$

Ces égalités découlent des identités trigonométriques

$$\sin^2\theta = \frac{1-\cos(2\theta)}{2},$$

$$\sum_{i=0}^{M-1}\cos(\alpha + i\delta) = \frac{\sin(M\delta/2)\cdot\cos(a+(M-1)\delta/2)}{\sin(\delta/2)}$$

et

$$\sin(A+B) - \sin(A-B) = 2\cos(A)\sin(B).$$

Si $M \geq 1/\sin(2\theta)$, alors

$$\frac{\sin(4M\theta)}{4M\sin(2\theta)} \leq \frac{1}{4M\sin(2\theta)} \leq \frac{1}{4}.$$

□

Ce lemme permet de prouver que l'algorithme d'amplification de l'amplitude, décrit ci-bas, trouve une solution à f avec un nombre de requêtes espéré dans l'ordre de $\sqrt{1/a}$. Cet algorithme prend comme paramètres la fonction f, l'opérateur G et une constante c qui servira de base à l'augmentation exponentielle de la valeur M.

AAmplif(f, G, c)

1. Fixer $\ell = 0$ et une constante $1 < c < 2$.
2. Incrémenter ℓ et fixer $M = \lceil c^\ell \rceil$.
3. Initialiser un registre quantique dans l'état $\mathsf{A}|0\rangle$.
4. Choisir j aléatoirement de manière uniforme entre 0 et $M - 1$.
5. Appliquer G^j au registre.
6. Mesurer le registre dans la base de calcul. Si le résultat x est une solution, s'arrêter et retourner x. Sinon, aller à l'étape 2.

Théorème 3. *Soit a la probabilité de mesurer une solution à f après avoir appliqué un algorithme quantique A, ne faisant aucune mesure, sur un état initial $|0\rangle$. L'algorithme* **AAmplif** *avec paramètres f, $\mathsf{G} = -\mathsf{A}\mathsf{S}_0\mathsf{A}^\dagger\mathsf{S}_f$ et $1 < c < 2$ trouve une solution à f avec un nombre de requête espéré dans l'ordre de $\sqrt{1/a}$.*

La preuve de cet théorème est retenue jusqu'au chapitre 4 où on traite de l'analyse de l'algorithme.

Pour le reste de ce document, nous confondrons le nombre de requêtes à f et le nombre d'applications de l'itération de Grover puisqu'un seul appel à f est nécessaire pour implémenter l'itération de Grover [BBC+95, BBHT98].

Chapitre 3

Tests de propriétés

Dans ce chapitre, on considère le problème de déterminer si une fonction boîte noire f est *près* d'avoir une certaine propriété ou doit être modifiée sur une proportion importante de ses entrées afin d'avoir cette propriété. Définissons d'abord la notion de distance.

Définition 4. *Soient $f : A \to B$ et $g : A \to B$ deux fonctions quelconques. On définit la* distance relative *entre f et g comme $d(f,g) = |\{x \in A \mid f(x) \neq g(x)\}|/|A|$. Soit $\mathcal{L}$ un ensemble de fonctions, la distance entre f et $\mathcal{L}$ est définie comme $d(f,\mathcal{L}) = \min_{g\in\mathcal{L}} d(f,g)$. On dit que f est ϵ-loin de $\mathcal{L}$ pour $\epsilon > 0$ si $d(f,\mathcal{L}) \geq \epsilon$ et que f est ϵ-près de $\mathcal{L}$ si elle n'est pas ϵ-loin de $\mathcal{L}$.*

La tâche que nous considérons est de *tester* si f a une propriété $\mathcal{L}$: sur entrée f, $\epsilon > 0$ et $\delta > 0$,

- accepter f avec probabilité au moins $1 - \delta$ si $f \in \mathcal{L}$ et
- rejeter f avec probabilité au moins $1 - \delta$ si $d(f,\mathcal{L}) \geq \epsilon$

avec la promesse que f satisfait un des deux cas. Le but est d'accomplir cette tâche avec le moins d'appels à f possible. Un algorithme testant f est appelé un *testeur* pour la propriété concernée.

3.1 Test de linéarité

La première propriété considérée est la linéarité de fonctions booléennes.

Définition 5. *Une fonction booléenne $f : \{0,1\}^n \to \{0,1\}$ est* linéaire *si*

$$f(x \oplus y) = f(x) \oplus f(y) \ \forall x, y \in \{0,1\}^n \tag{3.1}$$

où $\oplus$ est le OU-exclusif bit à bit.

Un testeur de linéarité doit alors, avec entrées f, $\epsilon > 0$ et $\delta > 0$, accepter f si elle est linéaire, la rejeter si elle est ϵ-loin de linéaire et se tromper avec probabilité au plus δ.

Classiquement, le meilleur algorithme connu est aussi le plus trivial. Découvert par Blum, Luby et Rubinfeld [BLR93], il consiste à répéter le test suivant : choisir $x, y \in_R \{0,1\}^n$ et vérifier si $f(x) \oplus f(y) = f(x \oplus y)$ jusqu'à ce que

1. on ait trouvé x, y tels que le test échoue, auquel cas on rejette f, ou
2. on ait effectué assez de tests pour être suffisamment certain que f est linéaire, auquel cas on accepte f.

Cet algorithme requiert un nombre d'appel à f dans l'ordre de $\frac{1}{\epsilon}$. En effet, si f est ϵ-loin de linéaire, il faut en moyenne faire un nombre de tests inversement proportionnel à ϵ avant qu'un test échoue. Il faut donc autant d'appels afin de pouvoir déclarer f linéaire et de ne se tromper qu'avec une probabilité bornée.

Lorsque l'on dispose de la mécanique quantique, la complexité de requête de ce problème peut être améliorée. L'algorithme classique correspond à la recherche d'une paire (x, y) exhibant la non-linéarité de f. Il s'agit là du genre de tâches où l'algorithme de Grover et ses généralisations excellent. Cependant, une méthode plus élégante peut être utilisée pour tester la linéarité de la fonction booléenne f. De plus, elle a l'avantage de retourner une description complète de f si celle-ci est linéaire. Cette méthode, utilisée par Hillery et Andersson dans [HA11], permet de résoudre le problème de tester la linéarité d'une fonction booléenne f avec un nombre d'appels à f dans $O(\epsilon^{-2/3} \log \frac{1}{\delta})$ pour un paramètre de distance ϵ et une probabilité d'erreur δ. Le reste de cette section présente un algorithme utilisant la même méthode, mais qui a les avantages d'être plus simple et de réduire le nombre de requêtes effectuées.

Une propriété des fonctions booléennes linéaires qui nous sera utile est qu'elles peuvent être représentées de matière succincte comme le produit scalaire d'une chaîne booléenne spécifique et de l'entrée à la fonction.

Propriété 6. *Soit $f : \{0,1\}^n \to \{0,1\}$ une fonction booléenne. La fonction f est linéaire si et seulement s'il existe $s \in \{0,1\}^n$ tel que $f(x) = s \cdot x \ \forall x \in \{0,1\}^n$ où $s \cdot x$ est le produit scalaire de deux chaînes booléennes.*

Démonstration. Si $f(x) = s \cdot x$ pour tout $x \in \{0,1\}^n$, alors $f(x \oplus y) = s \cdot (x \oplus y) = s \cdot x \oplus s \cdot y = f(x) \oplus f(y)$. Donc f est linéaire.

Si f est linéaire, alors soient $s_1 = f(10 \dots 0), s_2 = f(01 \dots 0), \dots, s_n = f(00 \dots 1)$. On doit avoir $f(0) = 0$, car $f(x) = f(x \oplus 0) = f(x) \oplus f(0)$. Donc, pour $x \in \{0,1\}^n$, on a $f(x) = f(x_1 x_2 \dots x_n) = f(x_1 0 \dots 0) \oplus f(0 x_2 \dots 0) \oplus \dots \oplus f(00 \dots x_n) = s_1 x_1 \oplus s_2 x_2 \oplus \dots \oplus s_n x_n = s \cdot x$ qui est le résultat voulu. □

L'utilité de cette propriété devient évidente lorsque l'on considère l'application de l'opérateur S_f à une superposition égale de tous les états de la base de calcul. Soit $f : \{0,1\}^n \to \{0,1\}$ une fonction booléenne linéaire, alors, par la propriété 6, il existe $s \in \{0,1\}^n$ tel que

$$\mathsf{S}_f \mathsf{H}^{\otimes n} |0\rangle = \mathsf{S}_f \frac{1}{2^{n/2}} \sum_x |x\rangle \tag{3.2}$$

$$= \frac{1}{2^{n/2}} \sum_x (-1)^{f(x)} |x\rangle \tag{3.3}$$

$$= \frac{1}{2^{n/2}} \sum_x (-1)^{s \cdot x} |x\rangle \tag{3.4}$$

$$= \mathsf{H}^{\otimes n} |s\rangle . \tag{3.5}$$

Appliquer l'opérateur auto-inverse $\mathsf{H}^{\otimes n}$ sur cet état et mesurer le registre retourne la chaîne s à coup sûr. On obtient donc une description complète de f en un seul appel à l'oracle S_f. Cette application de l'opérateur $\mathsf{H}^{\otimes n} \mathsf{S}_f \mathsf{H}^{\otimes n}$ suivi d'une mesure est appelée algorithme de Bernstein-Vazirani [BV97].

Supposons que l'on reçoive une fonction booléenne f et que l'on souhaite déterminer si f est linéaire ou ϵ-loin de linéaire. L'algorithme de Bernstein-Vazirani nous permet alors de définir une fonction candidate g qui possède les propriétés suivantes :

Définition 7. *Une fonction* candidate *pour f est une fonction g telle que*

- *g est linéaire ;*
- *si f est linéaire, alors $f = g$.*

Pour trouver un candidat pour f, remarquons que, lorsqu'appliquée sur une fonction f quelconque, la transformation $\mathsf{H}^{\otimes n}\mathsf{S}_f\mathsf{H}^{\otimes n}$ place l'état $|0\rangle$ dans une superposition $\sum_s \alpha_s |s\rangle$ où chaque chaîne s est en correspondance avec la fonction linéaire $g_s(x) = s{\cdot}x$. Mesurer le registre nous permet donc de définir une fonction avec les propriétés souhaitées. Le problème de tester la linéarité de f se réduit alors à tester l'égalité de f et sa fonction candidate g. Il suffit alors d'utiliser une procédure d'amplification de l'amplitude pour accomplir cette tâche.

L'algorithme complet, prenant comme paramètres la fonction $f\colon \{0,1\}^n \to \{0,1\}$, la distance $\epsilon > 0$ et l'erreur $\delta > 0$, procède ainsi :

LinTesteur(f, ϵ, δ)

1. Appliquer $\mathsf{H}^{\otimes n}\mathsf{S}_f\mathsf{H}^{\otimes n}$ à l'état $|0\rangle$ et mesurer. Soit s le résultat de la mesure.
2. Définir la fonction candidate g par $g(x) = s \cdot x$.
3. Définir la fonction booléenne χ par $\chi(x) = f(x) \oplus g(x)$. Définir les entiers $d = \lceil \log_{\frac{4}{3}} \frac{1}{\delta} \rceil$ et $M = \lceil \sqrt{1/\epsilon} \rceil$.
4. Choisir $j \in_{\mathrm{R}} [M]$, initialiser un registre dans l'état $\mathsf{H}|0\rangle$ et appliquer Q^j sur le registre où $\mathsf{Q} = \mathsf{H}^{\otimes n}\mathsf{S}_0\mathsf{H}^{\otimes n}\mathsf{S}_\chi$. Mesurer le registre. Si le résultat est une solution à χ, s'arrêter et rejeter f.
5. Répéter l'étape 4 d fois. Si aucune solution n'a été trouvée, accepter f.

Théorème 8. *Cet algorithme teste la linéarité de $f\colon \{0,1\}^n \to \{0,1\}$ avec un nombre de requêtes espéré dans $O(\log(\frac{1}{\delta})\sqrt{1/\epsilon})$ pour un paramètre de distance $\epsilon > 0$. Il se trompe avec probabilité au plus δ si f est ϵ-loin d'être linéaire et ne se trompe jamais si f est linéaire.*

Démonstration. L'algorithme de Bernstein-Vazirani retourne un candidat g pour f. Si f est linéaire, $f = g$ et cet algorithme accepte f car on ne trouve pas de solution à χ.

Si f est ϵ-loin de linéaire, alors comme g est linéaire, f est ϵ-loin de g. Soit $t = |\chi^{-1}(1)|$ le nombre de solutions de χ. On a $t = d(f,g) \geq \epsilon 2^n$. En choisissant $M = \lceil \sqrt{1/\epsilon} \rceil$, on a $M \geq \sqrt{2^n/t} \geq 1/\sin(2\theta)$ si $\theta \leq \pi/4$, ce que nous supposerons être le cas.

Le lemme 2 garantit que la probabilité de mesurer une solution à l'étape 4 est d'au moins $1/4$. En répétant d fois cette procédure, la probabilité de ne pas mesurer une solution décroit plus rapidement que $(\frac{3}{4})^d$. La probabilité d'erreur est alors bornée supérieurement par

$$(\frac{3}{4})^d \leq (\frac{3}{4})^{\log_{\frac{4}{3}} \frac{1}{\delta}} = (\frac{4}{3})^{-\log_{\frac{4}{3}} \frac{1}{\delta}} = (\frac{1}{\delta})^{-1} = \delta.$$

Chaque appel à χ requiert un seul appel à f car l'opérateur S_χ est simplement la composition des opérateurs pour g et f ; $\mathsf{S}_\chi = \mathsf{S}_f \mathsf{S}_g$. En tout, cet algorithme effectue au plus $O(dM) = O(\log(\frac{1}{\delta})\sqrt{1/\epsilon})$ appels à f. □

Cet algorithme utilise l'amplification de l'amplitude d'une nouvelle manière. Au lieu d'augmenter exponentiellement la valeur de M, on choisit une valeur pour laquelle on sait obtenir une bonne probabilité de succès. De cette manière, on évite les requêtes inutiles qui sont faites pendant les premières itérations de l'algorithme d'amplification de l'amplitude. Ce gain est possible grâce à la promesse des problèmes de test de propriété qui fait en sorte que s'il existe des solutions à l'oracle, alors nous avons une borne inférieure sur le nombre de ces solutions. Cette promesse sera toujours présente dans les prochaines sections de ce chapitre. Ainsi, les algorithmes présentés plus loin font aussi usage de cette nouvelle technique d'amplification de l'amplitude.

3.2 Test d'homomorphisme de groupes abéliens

La généralisation naturelle d'une fonction booléenne linéaire est un homomorphisme allant d'un groupe G vers un groupe H. Dans cette section, nous considérons les fonctions de la forme $f : G \to H$ où G et H sont des groupes abéliens finis. On souhaite alors tester si f est un homomorphisme, c'est-à-dire, si $f(x \oplus y) = f(x) \oplus f(y) \ \forall x, y \in G$, ou si f est ϵ-loin d'être un homomorphisme, où $\oplus$ dénote l'opération de groupe sur G ou H selon le contexte. Notons que l'algorithme classique présenté dans la section précédente s'applique également à ce type de fonction. Toutefois, la méthode quantique utilisée ne tient plus puisque l'algorithme de Bernstein-Vazirani ne s'applique pas à ce cas plus général.

L'algorithme quantique que nous présentons pour tester l'homomorphisme d'une fonction suit les mêmes grandes lignes que celui pour tester la linéarité de fonctions booléennes. Le premier outil requis pour résoudre ce problème est une généralisation de l'algorithme de Berstein-Vazirani aux fonctions sur groupes abéliens finis due à Høyer [Hø99].

Théorème 9 (Algorithme de Høyer)**.** *Soient G et H deux groupes abéliens finis et soit $f : G \to H$ un homomorphisme de groupe donné en boîte noire. Alors il existe un algorithme qui retourne une description complète de f en m appels à la boîte noire où m est la taille du plus petit ensemble qui génère H.*

Pour accomplir la même tâche classiquement, il faudrait évaluer f sur au moins n points de son domaine [Hø99] où n est la taille du plus petit ensemble qui génère G. Bien sûr, l'algorithme de Høyer n'est avantageux que si $m < n$, ce que nous supposerons être le cas, sans quoi l'algorithme classique trivial peut être utilisé.

Dans le contexte que nous considérons, f n'est pas toujours un homomorphisme tel que supposé par l'algorithme de Høyer. Toutefois, lorsque f n'est pas un homomorphisme, soit l'algorithme produira la description d'un homomorphisme g, soit un problème décelable surviendra durant l'exécution, ce qui nous permettra de conclure d'emblée que f est ϵ-loin d'un homomorphisme. Ainsi, dans ce qui suit, nous supposerons que l'algorithme de Høyer retourne toujours la description valide d'un homomorphisme. Il est important de noter que le nombre de requêtes qu'effectue l'algorithme de Høyer dépend seulement du codomaine de f et pas du fait que f soit ou non un homomorphisme.

L'algorithme de Høyer nous permet de trouver une fonction candidate g pour f. Il reste ensuite à tester l'égalité de f et g. Nous devons donc modifier de manière appropriée la fonction χ à laquelle nous voulons trouver une solution. Cette fois-ci le domaine de la fonction χ est le groupe G et elle prend ses valeurs dans $\{0, 1\}$. On définit $\chi : G \to \{0, 1\}$ par $\chi(x) = 1 \iff f(x) \neq g(x)$. Pour utiliser l'algorithme d'amplification de l'amplitude pour chercher une solution à χ, il faut implémenter l'opérateur d'inversion de la phase, S_χ. Pour ce faire, il suffit d'inverser la phase de l'état $|x\rangle$ lorsque $f(x) \oplus (-g(x)) \neq 0_H$ où $-g(x)$ dénote l'inverse de $g(x)$ dans H et 0_H est le neutre de H.

Pour chercher une solution à la nouvelle fonction χ, on procède presqu'exactement comme à la section précédente. Soit N le nombre d'éléments de G. Il suffit de remplacer, dans l'itération de Grover, $\mathsf{H}^{\otimes n}$ par la transformée de Fourier quantique F_N sur les éléments de G. Finalement, on rejette f si on trouve un élément $x \in G$ tel que $f(x) \neq g(x)$. Avec comme paramètres la fonction $f : G \to H$, la distance $\epsilon > 0$ et l'erreur $\delta > 0$, l'algorithme procède par les étapes suivantes :

HomTesteur(f, ϵ, δ)

1. Appliquer l'algorithme de Høyer. Soit g l'homomorphisme obtenu.
2. Définir la fonction booléenne χ par $\chi(x) = 1 \iff f(x) \neq g(x)$. Définir les entiers $d = \lceil \log_{\frac{4}{3}} \frac{1}{\delta} \rceil$ et $M = \lceil \sqrt{1/\epsilon} \rceil$.
3. Choisir $j \in_R [M]$, initialiser un registre dans l'état $\mathsf{F}_N |0\rangle$ et appliquer j fois l'opérateur $\mathsf{Q} = -\mathsf{F}_N \mathsf{S}_0 \mathsf{F}_N^\dagger \mathsf{S}_\chi$ sur le registre. Mesurer le registre. Si le résultat est une solution à χ, s'arrêter et rejeter f
4. Répéter l'étape 3 d fois. Si aucune solution n'a été trouvée, accepter f.

Théorème 10. *Soit m la taille du plus petit ensemble qui génère H. Cet algorithme teste si $f : G \to H$ est un homomorphisme avec un nombre de requêtes espéré dans $O(m + \log(\frac{1}{\delta})\sqrt{1/\epsilon})$ pour un paramètre de distance $\epsilon > 0$. Il se trompe avec probabilité au plus δ si f est ϵ-loin d'être un homomorphisme et ne se trompe jamais si f est un homomorphisme.*

Démonstration. Si f est un homomorphisme, alors l'étape 1 retourne une description complète de f et $f = g$. Dans ce cas, on ne trouvera pas de solution à χ, donc cet algorithme accepte f.

Si f est ϵ-loin d'être un homomorphisme, alors f est ϵ-loin de g. Il y a donc $t \geq \epsilon N$ solutions à χ. En applicant $j \in_R [M]$ fois l'opérateur Q pour $M = \lceil \sqrt{1/\epsilon} \rceil$, le lemme 2 nous garantit que mesurer donne une solution à χ avec probabilité au moins $1/4$ car $M \geq \sqrt{N/t} \geq 1/\sin(2\theta)$ si $\theta \geq \pi/4$, ce que nous supposons être le cas. En répétant $d = \lceil \log_{\frac{4}{3}} \frac{1}{\delta} \rceil$ fois, la probabilité de ne pas mesurer de solution est plus petite que $(\frac{1}{4})^d < \delta$. Ainsi, avec probabilité au moins $1 - \delta$, on rejette f.

Le nombre de requêtes à f est d'au plus m pour l'algorithme de Høyer et de $O(dM)$ pour l'amplification de l'amplitude. On effectue donc $O(m + \log(\frac{1}{\delta})\sqrt{1/\epsilon})$ requêtes au total. □

3.3 Test de symétrie

Dans cette section, nous considérons le problème de tester la symétrie d'une fonction booléenne.

Définition 11. *Une fonction* $f : \{0,1\}^n \to \{0,1\}$ *est* symétrique *si* $f(x) = f(y)$ *pour tout* $x, y \in \{0,1\}^n$ *tels que* $w(x) = w(y)$ *où* w*, le* poids de Hamming *d'une chaîne de bits, est le nombre de* 1 *dans cette chaîne.*

Autrement dit, f est symétrique si elle ne dépend que du nombre de 1 de son entrée. Une manière équivalente de dire que f est symétrique est de dire qu'elle est *invariante par permutation*, c'est-à-dire que $f(x_1 x_2 \ldots x_n) = f(x_{q(1)} x_{q(2)} \ldots x_{q(n)})$ pour toutes les permutations q des n premiers nombres naturels. Nous n'utiliserons toutefois que la première définition.

Un algorithme classique qui résout ce problème en $O(1/\epsilon)$ appels à f a été fourni par Majewski et Pippenger [MP09]. Il consiste à choisir $x \in_{\mathrm{R}} \{0,1\}^n$ et y aléatoirement parmi les chaînes de poids de Hamming $w(x)$, à vérifier que $f(x) = f(y)$ et à répéter jusqu'à ce que

1. on trouve x et y telles que le test échoue, auquel cas on rejette f ou
2. on a effectué assez de tests pour être suffisamment certain que f est symétrique, auquel cas on accepte f.

Dans le monde quantique, Hillery et Andersson [HA11] ont découvert une méthode permettant de tester la symétrie d'une fonction booléenne f avec $O(\epsilon^{-2/3})$ appels à f. Pour ce faire, ils testent l'appartenance du vecteur

$$|v_f\rangle = \mathsf{S}_f \mathsf{H}^{\otimes n} |0\rangle = \frac{1}{2^{n/2}} \sum_x (-1)^{f(x)} |x\rangle$$

au sous-espace symétrique, défini ci-dessous.

Définition 12. *Le* sous-espace symétrique $S \subset \mathcal{H}_2^{\otimes n}$ *est le sous-espace engendré par les vecteurs*

$$|u_i\rangle = \binom{n}{i}^{-\frac{1}{2}} \sum_{x\,:\,w(x)=i} |x\rangle$$

pour $i = 0, \ldots, n$.

Son complément orthogonal *est le sous-espace* $S^\perp = \{|\phi\rangle \mid \langle\phi \mid \psi\rangle = 0,\ \forall\, |\psi\rangle \in S\}$. *Ces sous-espaces satisfont la propriété* $\mathcal{H}_2^{\otimes n} = S \oplus S^\perp$.

Intuitivement, $|u_i\rangle$ est l'état consistant d'une superposition uniforme de tous les états de base avec poids de Hamming i. Si f est symétrique, le vecteur $|v_f\rangle$ réside entièrement dans S. L'importance de ce second sous-espace survient lorsque f est ϵ-loin de symétrique. Dans ce cas, le lemme suivant borne inférieurement la norme euclidienne d'une composante de $|v_f\rangle$ appartenant à $S^\perp$.

Lemme 13 ([HA11])**.** *Soient* $\mathsf{P}_{S^\perp}$ *la projection sur* $S^\perp$ *et* f *une fonction booléenne* ϵ*-loin de symétrique. Alors, la norme de la projection de* $|v_f\rangle$ *sur* $S^\perp$, $||\mathsf{P}_{S^\perp}\,|v_f\rangle\,||$, *est plus grande que* $\sqrt{2\epsilon}$. *Autrement dit,* $\langle v_f \mid \mathsf{P}_{S^\perp} \mid v_f\rangle \geq 2\epsilon$.

Grâce à ce résultat, une procédure semblable à celle utilisée dans la section 3.1 peut augmenter la probabilité de mesurer une composante de $|v_f\rangle$ résidant dans $S^\perp$. Il faut d'abord remarquer que, dans l'algorithme de l'amplification de l'amplitude présenté à la section 2.2, on peut modifier l'itération de Grover afin qu'elle augmente l'amplitude de la composante de $|v_f\rangle$ dans $S^\perp$. De plus, si on mesure selon l'observable $E_{\text{sym}} = \{S, S^\perp\}$ au lieu de la base de calcul, on peut arriver à mesurer une telle composante avec bonne probabilité en applicant l'itération modifiée un nombre de fois dans $O(\sqrt{1/\epsilon})$.

Pour ce faire, modifions l'itération de Grover de la manière suivante :

$$\mathsf{Q} = -\mathsf{A}\mathsf{S}_0\mathsf{A}^\dagger(\mathsf{I} - 2\mathsf{P}_S) \tag{3.6}$$

où $\mathsf{A} = \mathsf{S}_f\mathsf{H}^{\otimes n}$, et S_0 inverse la phase de l'état de base $|0\rangle$ et agît comme l'identité sur les autres états de base. L'action de cette nouvelle itération lorsqu'on mesure selon E_{sym}

s'apparente à celle de l'itération de Grover. C'est-à-dire, la probabilité de mesurer une composante dans $S_\perp$ est $\sin^2((2j+1)\theta)$ où θ est tel que $\sin^2(\theta) = \langle v_f \mid \mathsf{P}_{S^\perp} \mid v_f \rangle$. Ainsi, le lemme 2 tient toujours. En fait, retranscrivons-le en prenant en compte le nouveau contexte.

Lemme 14. *Soit $a = \langle v_f \mid \mathsf{P}_{S^\perp} \mid v_f \rangle$ la probabilité d'observer une composante dans $S^\perp$ lorsqu'on mesure $|v_f\rangle$ par rapport à l'observable E_{sym} et soit $0 < \theta < \pi/2$ tel que $\sin^2\theta = a$. Soient M un entier positif et j un entier choisi au hasard selon la distribution uniforme entre 0 et $M-1$. Après j applications de l'itération Q sur l'état initial $|v_f\rangle$, la probabilité de mesurer une solution est donnée par*

$$P_M = \frac{1}{2} - \frac{\sin(4M\theta)}{4M\sin(2\theta)}. \tag{3.7}$$

En particulier, $P_M \geq 1/4$ lorsque $M \geq 1/\sin(2\theta)$.

Ainsi, notre algorithme pour tester la symétrie de f est celui esquissé dans le dernier paragraphe :

SymTesteur(f, ϵ, δ)

1. Fixer $d = \lceil \log_{\frac{4}{3}} \frac{1}{\delta} \rceil$ et $M = \lceil \sqrt{1/\epsilon} \rceil$.
2. Initialiser un registre quantique dans l'état $\mathsf{A}\,|0\rangle = |v_f\rangle$.
3. Choisir j aléatoirement de manière uniforme entre 0 et $M-1$.
4. Appliquer Q^j au registre. Mesurer le registre selon l'observable E_{sym}. Si le résultat de la mesure est $S^\perp$, rejeter f.
5. Répéter d fois l'étape 4. Si aucune composante de $|v_f\rangle$ n'a été mesuré dans $S^\perp$, accepter f.

Théorème 15. *Cet algorithme teste la symétrie de $f\colon \{0,1\}^n \to \{0,1\}$ avec un nombre de requêtes espéré dans $O(\log(\frac{1}{\delta})\sqrt{1/\epsilon})$ pour un paramètre de distance $\epsilon > 0$. Il se trompe avec probabilité au plus δ si f est ϵ-loin d'être symétrique et ne se trompe jamais si f est symétrique.*

Démonstration. Si f est symétrique, alors $|v_f\rangle \in S$ et $a = 0$, on mesurera donc toujours S à l'étape 4. Cet algorithme accepte donc f.

Si f est ϵ-loin de symétrique, alors, par le lemme 13, la probabilité de mesurer $S^\perp$ est d'au moins 2ϵ. En choisissant $M = \lceil 1/\epsilon \rceil$, on a $M \geq \sqrt{1/a} \geq 1/\sin(2\theta)$ lorsque $\theta \leq \pi/4$, ce que nous supposons être le cas. Le lemme 14 garantit alors qu'on mesurera $S^\perp$ avec probabilité au moins $1/4$ à l'étape 4. Ainsi, en répétant $d = \lceil \log_{\frac{4}{3}} \frac{1}{\delta} \rceil$ fois, la probabilité de ne pas mesurer $S^\perp$ est d'au plus $(\frac{3}{4})^d < \delta$. Au final, on rejette f avec probabilité au moins $1 - \delta$.

Le nombre d'applications de l'opérateur Q est d'au plus dM. Ainsi, le nombre d'appels à f est dans $O(\log \frac{1}{\delta} \sqrt{1/\epsilon})$. □

3.4 Test de quasi-symétrie

Dans cette section, nous considérons une propriété plus générale que la symétrie d'une fonction booléenne qui est celle d'être symétrique sur les bits dont la fonction dépend.

Définition 16. *Une fonction* $f : \{0,1\}^n \to \{0,1\}$ dépend *de son* i^e *bit s'il existe des bits* $x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}$ *tels que*

$$f(x_0 \ldots x_{i-1} 0 x_{i+1} \ldots x_{n-1}) \neq f(x_0 \ldots x_{i-1} 1 x_{i+1} \ldots x_{n-1}).$$

Soient $J \subseteq [n]$ *un ensemble d'indices et* $z \in \{0,1\}^n$. *La fonction booléenne* $f_z : \{0,1\}^{|J|} \to \{0,1\}$ *qui est telle que*

$$f_z(x) = f(x') \quad \textit{où} \begin{cases} x'_i = x_i & \textit{si } i \in J \textit{ et} \\ x'_i = z_i & \textit{sinon} \end{cases}$$

est une restriction *de* f *aux indices dans* J. *On dit que* f *est* quasi-symétrique *si la restriction de* f *aux indices dont elle dépend est symétrique.*

Dans la définition précédente, les bits de f_z sont indexés selon les éléments de J. Si f dépend d'indices en dehors de J, sa restriction f_z n'est pas nécessairement unique

et dépend du choix de z. La restriction de f aux indices dont elle dépend est toujours unique.

Un algorithme classique testant la quasi-symétrie d'une fonction booléenne est connu. Il est dû à Majewski et Pippenger [MP09]. Il *estime* d'abord l'ensemble des bits desquels la fonction dépend, puis applique l'algorithme de test de symétrie sur des restrictions de la fonction aux bits de l'ensemble trouvé à la première étape. Il fait un nombre de requêtes dans $O(\log(\frac{n}{\delta})\frac{n}{\epsilon})$. Notre algorithme suit cette structure ; un algorithme quantique estimant les bits dont f dépend est d'abord présenté à la section 3.4.1, puis à la section suivante, l'algorithme complet.

3.4.1 Estimation de la dépendance

Définissons d'abord rigoureusement ce qu'est un estimé de l'ensemble des bits dont une fonction dépend.

Définition 17. *Soient $f : \{0,1\}^n \to \{0,1\}$ une fonction booléenne et $\epsilon > 0$. Soit $\mathcal{J}(f)$ l'ensemble des indices des bits dont dépend f. Un ϵ-estimé pour $\mathcal{J}(f)$ est un ensemble $J \subseteq [n]$ tel que*

1. *$J \subseteq \mathcal{J}(f)$ et*
2. *f est ϵ-près de l'ensemble des fonctions dépendant seulement des indices dans J, c'est-à-dire $d(f, \mathcal{J}^{-1}(J)) \leq \epsilon$.*

Pour estimer la dépendance de f, nous aurons besoin d'un outil qui nous permet de trouver l'indice d'un bit dont f dépend.

Lemme 18 ([MP09])**.** *Soient x et y tels que $f(x) \neq f(y)$. Il existe une procédure* **RechDep** *prenant comme paramètres x et y et qui retourne l'indice d'un bit duquel f dépend en $O(\log n)$ appels à f.*

Nous exhibons cet algorithme :

RechDep(x, y, f)

1. Si $w(x \oplus y) = 1$, retourner l'unique i tel que $x_i \neq y_i$.
2. Sinon, soit $z \in \{0,1\}^n$ tel que $w(x \oplus z) \leq \lceil w(x \oplus y)/2 \rceil$ et $w(y \oplus z) \leq \lceil w(x \oplus y)/2 \rceil$.
3. Si $f(x) \neq f(z)$, retourner **RechDep**(x, z, f). Sinon, retourner **RechDep**(z, y, f).

À chaque appel récursif, la valeur $|x \oplus y|$ est *environ* réduite de moitié. Cet algorithme s'apparente à une fouille dichotomique. Pour cette raison, nombre total de requêtes à f est au plus $O(\log n)$.

Grâce à ce lemme, il nous suffit de trouver deux entrées sur lesquelles f ne prend pas la même valeur pour identifier efficacement un bit duquel elle dépend.

Considérons, par abus de notation, une restriction f_z de f à un ensemble J comme une fonction de n bits (au lieu de $|J|$ bits) en ignorant les indices en dehors de J. Si f est ϵ-loin des fonctions qui dépendent de J, alors f est aussi ϵ-loin de sa restriction f_z. Il existe donc au moins $\epsilon 2^n$ éléments pour lesquels f et f_z prennent une valeur différente. On peut donc utiliser l'amplification de l'amplitude pour trouver de tels éléments. À chaque fois qu'on en trouve un, on identifie du même coup un bit supplémentaire dont f dépend. Cette idée est formalisée dans l'algorithme suivant :

$\mathbf{EstDep}(f, \epsilon, \delta)$

1. Définir $J = \emptyset$ et $\delta' = \delta/n$.
2. Soit $z \in_R \{0,1\}^n$. Définir f_z telle que $f_z(x) = f(y)$ où y est tel que $y_i = x_i$ si $i \in J$ et $y_i = z_i$ sinon.
3. Utiliser l'algorithme d'amplification de l'amplitude pour trouver une solution à $\chi(x) = f(x) \oplus f_z(x)$.
   (a) Définir $\mathsf{Q} = \mathsf{H}^{\otimes n}\mathsf{S}_0\mathsf{H}^{\otimes n}\mathsf{S}_\chi$, $d = \lceil \log_{4/3}(1/\delta') \rceil$ et $M = \lceil \sqrt{1/\epsilon} \rceil$.
   (b) Choisir $j \in_R [M]$, appliquer Q^j sur l'état $\mathsf{H}|0\rangle$ et mesurer.
   (c) Répéter l'étape 3b d fois ou jusqu'à ce qu'on mesure une solution.
4. Si une solution x a été trouvée, calculer $i = \mathbf{RechDep}(x, y, f)$ où y est la chaîne obtenue à partir de x comme à l'étape 2. Mettre à jour $J \leftarrow J \cup \{i\}$ et retourner à l'étape 2.
5. Sinon, retourner J.

Théorème 19. *L'algorithme* $\mathbf{EstDep}$ *retourne un* ϵ*-estimé de* $\mathcal{J}(f)$ *avec probabilité au moins* $1 - \delta$ *en* $O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\delta})$ *requêtes.*

Démonstration. Soit J la sortie de l'algorithme. Il est clair que $J \subseteq \mathcal{J}(f)$ car les indices retournés par $\mathbf{RechDep}$ sont seulement ceux dont f dépend. La première propriété d'un ϵ-estimé pour $\mathcal{J}(f)$ est donc satisfaite.

Pour montrer que la deuxième propriété est également satisfaite, supposons qu'à l'étape 3 f est ϵ-loin des fonctions qui dépendent seulement des indices dans J. Alors puisque f est aussi ϵ-loin de f_z, l'étape 3b trouve une solution à la fonction χ avec probabilité au moins $1/4$ par le lemme 2. Ainsi, avec probabilité au moins $1-(3/4)^d \geq 1-\delta'$, la procédure d'amplification de l'amplitude trouve une solution. Lorsque c'est le cas, l'algorithme fait un tour de boucle supplémentaire et ajoute un indice à J. Comme au plus un indice s'ajoute à J à chaque tour de boucle, l'algorithme effectue au plus n itérations. Chacune d'entre elles échoue avec probabilité au plus $\delta' = \delta/n$, donc, par l'inégalité

de Boole, la probabilité qu'au moins une d'entre elles échoue est d'au plus $n\delta' = \delta$. On satisfait donc la deuxième propriété avec probabilité au moins $1 - \delta$.

La recherche d'une solution à χ prend au plus dM appels à χ. Trouver un bit dont f dépend à partir de cette solution prend $O(\log n)$ appels à f. Ce qui précède est répété au plus pour chaque bit dont f dépend, soit au plus n fois. Au total, le nombre de requêtes effectuées est dans $O(n(\frac{1}{\sqrt{\epsilon}} \log \frac{n}{\delta} + \log n)) = O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\delta})$. □

3.4.2 L'algorithme

Tel que mentionné au début de cette section, notre testeur de quasi-symétrie estimera d'abord les indices sur lesquels dépend la fonction testée puis appliquera l'algorithme pour tester la symétrie à une fonction restreinte aux indices dont elle dépend. La conformité de notre algorithme, présenté plus loin, découle du lemme 21, dû à Majewski et Pippenger [MP09]. Mais avant de l'énoncer, exhibons une propriété de la mesure de distance définie en début de chapitre.

Propriété 20. *Soit $d(\cdot, \cdot)$ la distance de la définition 4. Cette distance satisfait toutes les propriétés d'une métrique sur l'ensemble $\mathcal{B}_n$ des fonctions booléennes à n bits d'entrée, c'est-à-dire qu'elle est nulle seulement si les fonctions sont égales, elle est symétrique et satisfait l'inégalité du triangle. De plus, si on pose $d(f, g) = \infty$ pour $f \in \mathcal{B}_n$, $g \in \mathcal{B}_m$ avec $m \neq n$, alors d est aussi une métrique sur l'ensemble $\mathcal{B} = \bigcup_n \mathcal{B}_n$.*

Lemme 21 ([MP09])**.** *Soit $f : \{0,1\}^n \to \{0,1\}$ et soit J un ϵ-estimé des indices dont dépend f. Si f est 4ϵ-loin d'être quasi-symétrique, alors tout testeur de symétrie avec paramètres ϵ et δ rejette f_z, une restriction de f à J avec un z aléatoire uniformément distribué, avec probabilité au moins $\frac{1}{2}(1 - \delta)$.*

Démonstration. Notons $\mathcal{S} \subset \mathcal{B}$ l'ensemble des fonctions symétriques et $\mathcal{Q} \subset \mathcal{S}$ l'ensemble des fonctions quasi-symétriques.

Soit $g : \{0,1\}^n \to \{0,1\}$ une fonction qui dépend seulement d'indices dans J et qui minimise $d(f, g)$. Alors, puisque J est un ϵ-estimé de $\mathcal{J}(f)$, on a $d(f, g) \leq \epsilon$. Ainsi, l'inégalité du triangle $d(f, g) + d(g, \mathcal{Q}) \geq d(f, \mathcal{Q}) \geq 4\epsilon$ implique que $d(g, \mathcal{Q}) \geq 3\epsilon$.

Soit alors g_z la restriction de g aux indices dans J. Comme g ne dépend que d'indices dans J et comme $\mathcal{S} \subset \mathcal{Q}$, l'inégalité $d(g_z, \mathcal{S}) \geq d(g_z, \mathcal{Q}) \geq d(g, \mathcal{Q}) \geq 3\epsilon$ tient. Considérons la distance $d(f_z, g_z)$ entre f_z et g_z. Cette distance est une variable aléatoire car f peut dépendre d'indices en dehors de J. L'espérance de $d(f_z, g_z)$ est $d(f, g)$. Ainsi, par l'inégalité de Markov, on a $d(f_z, g_z) \leq 2d(f, g) \leq 2\epsilon$ avec probabilité au moins $1/2$. Lorsque c'est le cas, l'inégalité $d(f_z, g_z) + d(f_z, \mathcal{S}) \geq d(g_z, \mathcal{S}) \geq 3\epsilon$ nous garantit que $d(f_z, \mathcal{S}) \geq \epsilon$ et que le testeur de symétrie rejette f_z avec probabilité au moins $1 - \delta$. En général, le testeur de symétrie rejette f_z avec probabilité au moins $\frac{1}{2}(1-\delta)$ ce qui prouve le résultat. □

Ce lemme tient pour tout testeur de symétrie satisfaisant la définition donnée en début de chapitre. Notre testeur de symétrie $\mathbf{SymTesteur}$ a la propriété supplémentaire qu'il accepte toujours la fonction f si celle-ci est symétrique. Cette propriété permettra d'amplifier la probabilité de succès donnée par le lemme précédent.

L'algorithme de test de quasi-symétrie est donné ci-bas. Il estime la dépendance de f, puis teste la symétrie de restrictions de f à l'ensemble d'indices trouvés. Le lemme précédent nous permet de faire des choix appropriés pour les sous-appels faits dans l'algorithme.

QSymTesteur(f, ϵ, δ)

1. Poser $J = \mathbf{EstDep}(\epsilon/4, \delta/2)$.
2. Soit $k = \lceil \log_{3/2}(2/\delta) \rceil$.
3. Générer une restriction f_z de f à J et appeler l'algorithme $\mathbf{SymTesteur}(f_z, \epsilon/4, 1/3)$. Si le testeur de symétrie a rejeté f_z, rejeter f.
4. Répéter k fois l'étape 3.
5. Si aucun des tests de symétrie n'a échoué, accepter f.

Théorème 22. *Cet algorithme teste la quasi-symétrie de* $f : \{0,1\}^n \to \{0,1\}$ *avec un nombre de requêtes espéré dans* $O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\delta})$ *pour un paramètre de distance* $\epsilon > 0$ *et se*

trompe avec probabilité au plus δ si f est ϵ-loin d'être quasi-symétrique et ne se trompe jamais si f est quasi-symétrique.

Démonstration. Si f est quasi-symétrique, f_z est symétrique et chaque test de symétrie réussit.

Supposons que J soit un $\epsilon/4$-estimé de $\mathcal{J}(f)$. Si f est ϵ-loin de symétrique, par le lemme 21, chaque test de symétrie rejette f_z avec probabilité au moins $\frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$. Donc la probabilité que tous les tests acceptent f_z est d'au plus $(\frac{2}{3})^k \leq \delta/2$. Ainsi on accepte f avec probabilité inférieure à $\frac{\delta}{2} + \frac{\delta}{2} = \delta$, soit la probabilité d'échouer l'estimé de $\mathcal{J}(f)$ plus la probabilité que tous les tests de symétrie acceptent f_z. Dans tous les cas, l'algorithme effectue $O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\delta}) + k \cdot O(\sqrt{\frac{1}{\epsilon}}) = O(\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\delta})$ requêtes. □

Chapitre 4

Analyse de l'algorithme d'amplification de l'amplitude

L'algorithme d'amplification de l'amplitude, décrit à la section 2.2, a été largement étudié et ses applications sont nombreuses comme en témoignent les sections précédentes. Le nombre de requêtes à la boîte noire faites par cet algorithme est une variable aléatoire prenant sa source d'aléa dans l'incertitude associée à la mesure d'un état quantique. On veut donc connaître le nombre de requêtes espéré que l'algorithme $\mathbf{AAmplif}$ effectue. Cette valeur a déjà été étudiée de manière exhaustive [BBHT98, BHMT02]. Toutefois, le comportement de la variance du nombre de requêtes effectuées par l'algorithme reste méconnue. Cette mesure de la dispersion des différentes réalisations du nombre de requêtes influe sur les utilisations possibles de cet algorithme. Lorsque la variance est petite, c'est un gage que la répartition du nombre de requêtes est près de la moyenne, une propriété souhaitable. Avec une très grande variance, les évènements rares qui correspondent aux exécutions de l'algorithme dépassant de beaucoup le nombre espéré de requêtes sont plus fréquents.

4.1 La distribution du nombre de requêtes

Dans cette section est présenté la distribution de probabilité du nombre de requêtes qu'effectue l'algorithme d'amplification de l'amplitude, décrit à la section 2.2. Ce nombre est en partie déterminé par le moment auquel on mesure une solution à l'oracle. Le nombre aléatoire d'itérations de Grover qui sont effectuées à chaque itération de l'algorithme est ce qui, ultimement, s'additionnera pour donner le nombre total de requêtes.

L'algorithme d'amplification de l'amplitude s'arrête lorsqu'il a trouvé une solution. Pour connaître le nombre total d'itérations que fait l'algorithme, il faut d'abord savoir quand il s'arrête. La probabilité de mesurer une solution à une itération ℓ de $\mathbf{AAmplif}$ correspond à la probabilité de mesurer une solution en applicant un nombre aléatoire de fois l'itération de Grover. Cette probabilité est donnée dans le lemme 2 par la formule

suivante :

$$P_M = \frac{1}{2} - \frac{\sin(4M\theta)}{4M\sin(2\theta)}$$

où $M = \lceil c^\ell \rceil$ est la valeur maximale d'itérations de Grover qui peuvent être appliquées. Soit L la variable aléatoire du nombre d'itérations effectuées. Elle prend ses valeurs dans les nombres naturels positifs. Ainsi, la probabilité que $L = 1$ est égale à la probabilité de mesurer une solution à la première itération, la probabilité que $L = 2$ est égale à la probabilité que la première itération échoue et qu'on mesure une solution à la deuxième itération, etc. On peut donc exprimer la fonction de masse de L par

$$\Pr[L = \ell] = P_{\lceil c^\ell \rceil} \prod_{i=1}^{\ell-1} (1 - P_{\lceil c^i \rceil})$$

pour $\ell \in \mathbb{N}$.

Le nombre aléatoire de requêtes qu'effectue l'algorithme est simplement une somme de variables aléatoires uniformes. En effet, soit $M_i = \lceil c^i \rceil$, à chaque itération de l'algorithme, on choisit avec probabilité $1/M_i$ de faire $j \in [M_i]$ applications de l'itération de Grover. Soit $U_i \sim \text{Uniforme}(0, M_i - 1)$ le nombre aléatoire de requêtes faites à la i^{e} itération de l'algorithme d'amplification de l'amplitude, alors le nombre total d'appels à l'oracle est donné par la variable aléatoire

$$X = \sum_{i=1}^{L} U_i$$

où la somme est sur un nombre aléatoire de termes.

Par simplicité, on considère que la valeur M peut prendre des valeurs autres qu'entières, c'est-à-dire qu'on prend $M = c^i$ pour l'itération i de l'algorithme $\mathbf{AAmplif}$. Les U_i seront alors des variables aléatoires uniformes continues allant de 0 à c^i. Ceci a un impact négligeable sur les quantités calculées plus loin.

On voit donc que le nombre total de requêtes de l'algorithme dépend grandement du nombre d'itérations que celui-ci effectue. Comme il est très aisé de travailler avec des

variables uniformes, la difficulté de l'analyse résidera dans la distribution de probabilité de la variable aléatoire L.

Les lois suivantes, que l'on retrouve dans [WHH06], nous seront utiles pour l'analyse de l'espérance et de la variance. La première, la *loi de l'espérance itérée* stipule que

$$\mathrm{E}_X[X] = \mathrm{E}_L[\mathrm{E}_X[X \mid L]]$$

où $\mathrm{E}_X[X \mid L = \ell]$ est l'espérance conditionnelle qui, sur paramètre ℓ, prend la valeur

$$\mathrm{E}_X[X \mid L = \ell] = \sum_x x \cdot \Pr[X = x \mid L = \ell].$$

La deuxième, la *loi de la variance totale*, affirme que

$$\mathrm{Var}_X[X] = \mathrm{E}_L[\mathrm{Var}_X[X \mid L]] + \mathrm{Var}_L[\mathrm{E}_X[X \mid L]].$$

où $\mathrm{Var}_X[X \mid L = \ell]$ est la variance conditionnelle définie par

$$\begin{aligned}\mathrm{Var}_X[X \mid L = \ell] &= \mathrm{E}[(X - \mathrm{E}[X \mid L = \ell])^2 \mid L = \ell] \\ &= \mathrm{E}[X^2 \mid L = \ell] - (\mathrm{E}[X \mid L = \ell])^2\end{aligned}$$

Nous calculons les deux valeurs suivantes qui nous servirons par la suite.

Lemme 23. *L'espérance de X conditionnelle à $L = \ell$ est*

$$\mathrm{E}[X \mid L = \ell] = \frac{c^{\ell+1} - c}{2(c-1)} \in \Theta(c^\ell)$$

et la variance de X conditionnelle à $L = \ell$ est

$$\mathrm{Var}[X \mid L = \ell] = \frac{c^{2\ell+2} - \ell - c^2}{12(c^2-1)} \in \Theta(c^{2\ell}).$$

Démonstration. On obtient les deux valeurs par les développements suivants :

$$\begin{aligned}
\mathrm{E}[X \mid L = \ell] &= \mathrm{E}[\sum_{i=1}^{L} U_i \mid L = \ell] \\
&= \sum_{i=1}^{\ell} \mathrm{E}[U_i] \\
&= \sum_{i=1}^{\ell} \frac{c^i}{2} \\
&= \frac{c^{\ell+1} - c}{2(c-1)}
\end{aligned}$$

et

$$\begin{aligned}
\mathrm{Var}[X \mid L = \ell] &= \mathrm{Var}[\sum_{i=1}^{L} U_i \mid L = \ell] \\
&= \sum_{i=1}^{\ell} \mathrm{Var}[U_i] \\
&= \sum_{i=1}^{\ell} \frac{c^{2i} - 1}{12} \\
&= \frac{c^{2\ell+2} - \ell - c^2}{12(c^2-1)}
\end{aligned}$$

où les U_i sont des variables aléatoires uniformes continues allant de 0 à c^i. On a fait usage du fait que la variance d'une somme de variables aléatoires indépendantes est égale à la somme des variances de ces variables aléatoires. Plus formellement :

$$\mathrm{Var}\left[\sum_i U_i\right] = \sum_i \mathrm{Var}[U_i]$$

lorsque $\mathrm{Cov}[U_i, U_j] = 0$ pour $i \neq j$. □

4.2 L'espérance

L'espérance du nombre d'appels à l'oracle est la propriété la plus importante de cette variable aléatoire. C'est elle qui décrit le comportement moyen de plusieurs appels à l'algorithme. Cette valeur a été bornée supérieurement par Brassard, Høyer, Mosca et Tapp [BHMT02] dans le théorème 3. La preuve de leur résultat est présentée ici.

Théorème 3. *Soit a la probabilité de mesurer une solution à f après avoir appliqué un algorithme quantique A, ne faisant aucune mesure, sur un état initial $|0\rangle$. L'algorithme* **AAmplif** *avec paramètres f, $\mathsf{G} = -\mathsf{A}^{-1}\mathsf{S}_0\mathsf{A}\mathsf{S}_f$ et $1 < c < 2$ trouve une solution à f avec un nombre de requête dans l'ordre de $\sqrt{1/a}$.*

Démonstration du théorème 3. L'espérance du nombre de requêtes faites par l'algorithme d'amplification de l'amplitude est

$$\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}_L[\mathrm{E}_X[X \mid L]] \\
&= \sum_{\ell=1}^{\infty} \Pr[L = \ell]\mathrm{E}_X[X \mid L = \ell] \\
&= \sum_{\ell=1}^{\infty} \Pr[L = \ell]\frac{c^{\ell+1} - c}{2(c-1)}
\end{aligned}$$

Cette série est positive à condition que $c > 1$. À une constante multiplicative près, cette série est bornée supérieurement par

$$\sum_{\ell=1}^{\infty} \Pr[L = \ell]c^{\ell} \tag{4.1}$$

Le lemme 2 permet de borner inférieurement la probabilité de mesurer une solution par

$$P_M = \frac{1}{2}\left(1 - \frac{1}{2M\sqrt{a}}\right).$$

Soient $0 < c_0 < 1/2$ tel que $c = 2(1 - c_0)$ et $M_0 = 1/(2c_0\sqrt{a}\,)$. Le fait que c soit plus petit que 2 nous assure que $M_0 > 0$. On peut alors séparer la série (4.1) en deux :

celle contenant les termes allant de 1 à $\lceil \log_c M_0 \rceil$ et celle contenant les termes restants. La première série est dans $O(M_0)$ car

$$\begin{aligned}\sum_{\ell=1}^{\lceil \log_c M_0 \rceil} \Pr[L = \ell] c^\ell &\leq \sum_{\ell=1}^{\lceil \log_c M_0 \rceil} c^\ell \\ &= \frac{c^{\lceil \log_c M_0 \rceil + 1} - c}{c - 1} \\ &\in O(M_0)\end{aligned}$$

Pour les termes restants de la série, on remarque que si $M > M_0$, alors $P_M \geq \frac{1}{2}(1 - c_0)$. Ainsi, pour chaque itération $\ell \geq \lceil \log_c M_0 \rceil$, on se rendra à l'itération $\ell + 1$ avec probabilité au plus $p_0 = 1 - P_{M_0} = \frac{1}{2}(1 + c_0)$. La seconde série peut donc se simplifier en

$$\begin{aligned}\sum_{\ell=\lceil \log_c M_0 \rceil + 1}^{\infty} \Pr[L = \ell] c^\ell &= \sum_{\ell=\lceil \log_c M_0 \rceil + 1}^{\infty} P_{\lceil c^\ell \rceil} \prod_{i=1}^{\ell-1} (1 - P_{\lceil c^i \rceil}) c^\ell \\ &\leq \sum_{\ell=1}^{\infty} (p_0)^\ell c^{\ell + \lceil \log_c M_0 \rceil + 1} \\ &\approx \sum_{\ell=1}^{\infty} M_0 (c p_0)^\ell.\end{aligned}$$

Cette série est dans $O(M_0)$ car $cp_0 = 2(1 - c_0)\frac{1}{2}(1 + c_0) = 1 - c_0^2 < 1$. Ainsi le nombre espéré de requêtes est dans $O(M_0) = O(\sqrt{1/a})$. □

L'espérance du nombre de requêtes faites par l'algorithme d'amplification de l'amplitude est donc du même ordre de grandeur que celui lorsque la probabilité de succès a est connue.

4.3 La variance

Dans cette section, nous nous intéressons à la variance du nombre de requêtes qu'effectue l'algorithme d'amplification de l'amplitude. Cette étude est motivée entres autres par le fait que certaines applications que l'on pourrait faire de cet algorithme demandent

à celui-ci d'avoir un comportement près de la moyenne.

Dans l'algorithme d'amplification de l'amplitude, le facteur qui influence le plus la variance du nombre de requêtes est la valeur choisie pour le paramètre c. Celle-ci prend ses valeurs strictement entre 1 et 2. Cet intervalle garantit que l'espérance du nombre d'appels à l'oracle est fini, mais comme le montre le prochain théorème, cela ne suffit pas pour borner la variance.

Le premier résultat original de cette section montre que, pour certaines valeurs de c dans l'intervalle $]1, 2[$, la variance est infinie.

Théorème 24. *Soit c le paramètre dans l'algorithme d'amplification de l'amplitude. Si $c > \sqrt{2}$, alors la variance du nombre de requêtes est infinie.*

Démonstration. Par la loi de la variance totale, la variance de X est

$$\begin{aligned}\mathrm{Var}[X] &= \mathrm{E}_L[\mathrm{Var}_X[X \mid L]] + \mathrm{Var}_L[\mathrm{E}_X[X \mid L]] \\ &\geq \mathrm{E}_L[\mathrm{Var}_X[X \mid L]]\end{aligned}$$

Par le lemme 23, cette valeur est

$$\begin{aligned}\mathrm{E}_L[\mathrm{Var}_X[X \mid L]] &= \sum_{\ell=1}^{\infty} \Pr[L = \ell]\mathrm{Var}_X[X \mid L = \ell] \\ &= \sum_{\ell=1}^{\infty} \Pr[L = \ell]\frac{c^{2\ell+2} - \ell - c^2}{12(c^2-1)}.\end{aligned}$$

Puisque $\frac{c^{2\ell+2}-\ell-c^2}{12(c^2-1)} \in \Omega\left(c^{2\ell}\right)$, cette série est plus grande ou égale à

$$k \cdot \sum_{\ell=\ell_0}^{\infty} \Pr[L = \ell]c^{2\ell} \tag{4.2}$$

pour des constantes k et ℓ_0 associées à la notation Ω.

Tout comme la probabilité de mesurer une bonne solution à chaque itération, la probabilité de ne pas mesurer de solution tend vers $1/2$ lorsque ℓ augmente. En effet, par le

lemme 2, lorsque $M \geq s/\sin(2\theta)$ pour $s \in \mathbb{N}$, on a

$$\begin{aligned} 1 - P_M &= \frac{1}{2} + \frac{\sin(4M\theta)}{4M\sin(2\theta)} \\ &\geq \frac{1}{2} - \frac{1}{4M\sin(2\theta)} \\ &\geq \frac{1}{2} - \frac{1}{4s} \\ &= \frac{2 - 1/s}{4}. \end{aligned}$$

Il existe alors s tel que $\frac{2-1/s}{4}c^2 > 1$. Notamment, pour $\epsilon > 0$ tel que $c^2 = 2 + \epsilon$, on a $\frac{2-1/s}{4}c^2 > 1$ pour $s > \frac{1}{\epsilon} + \frac{1}{2}$. Soient alors $M = s/\sin(2\theta)$ et $\ell_M = \lceil \log_c M \rceil$. La série (4.2), à une constante multiplicative près, est plus grande[1] que

$$\sum_{\ell=\ell_M}^{\infty} \left(\frac{2-1/s}{4}\right)^{\ell-\ell_M} c^{2\ell} = c^{2\ell_M} \cdot \sum_{\ell'=0}^{\infty} \left(\frac{2-1/s}{4}\right)^{\ell'} c^{2\ell'}$$

par le changement de variable $\ell' = \ell - \ell_M$. Cette série diverge car $\frac{2-1/s}{4}c^2 > 1$. □

Par opposition au résultat précédent, le prochain montre que la variance peut être bornée supérieurement pour l'autre partie des valeurs possibles de c.

Théorème 25. *La variance du nombre de requêtes faites par l'algorithme d'amplification de l'amplitude est finie pour* $1 < c < \sqrt{2}$.

Démonstration. En utilisant la loi de la variance totale et le lemme 23,

$$\begin{aligned} \mathrm{Var}[X] &= \mathrm{E}_L[\mathrm{Var}_X[X \mid L]] + \mathrm{Var}_L[\mathrm{E}_X[X \mid L]] \\ &= \mathrm{E}_L[\mathrm{Var}_X[X \mid L]] + \mathrm{E}_L[(\mathrm{E}_X[X \mid L])^2] - (\mathrm{E}_L[\mathrm{E}_X[X \mid L]])^2 \\ &= \sum_{\ell=\ell_0}^{\infty} \Pr[L = \ell] \frac{c^{2\ell+2} - \ell - 1}{12(c^2-1)} + \sum_{\ell=\ell_0}^{\infty} \Pr[L = \ell] \left(\frac{c^{\ell+1} - c}{2(c-1)}\right)^2 - (\mathrm{E}[X])^2. \end{aligned}$$

Puisqu'on cherche à borner supérieurement, on peut ignorer le dernier terme. Les deux autres sont des séries infinies de termes dans $O(c^{2\ell})$. Elles peuvent donc être bornées

[1] On suppose s.p.d.g. que $\ell_0 \leq \ell_M$.

supérieurement, à une constante près, par

$$\sum_{\ell=1}^{\infty} \Pr[L = \ell] c^{2\ell}. \tag{4.3}$$

Pour tenter de borner supérieurement $\Pr[L = \ell]$, remarquons que la probabilité P_M de mesurer une bonne solution en applicant un nombre d'itérations de Grover aléatoire entre 0 et $M - 1$ est telle que

$$P_M \geq \frac{1}{2}\left(1 - \frac{1}{2M\sqrt{a}}\right)$$

ce qui implique que la probabilité de mesurer un mauvaise solution est

$$1 - P_M \leq \frac{1}{2}\left(1 + \frac{1}{2M\sqrt{a}}\right).$$

Soit $M_0 = \frac{c^2}{2(2-c^2)\sqrt{a}}$ la plus petite solution à $\frac{1}{2}\left(1 + \frac{1}{2M\sqrt{a}}\right) c^2 < 1$ lorsque $0 < c < \sqrt{2}$. On peut diviser la série (4.3) en deux : l'une correspondant aux termes pour lesquels $M \leq M_0$ et l'autre pour $M > M_0$. Elle devient alors, à une constante multiplicative près

$$\sum_{\ell=1}^{\lceil \log_c M_0 \rceil} \Pr[L = \ell] c^{2\ell} + \sum_{\ell=\lceil \log_c M_0 \rceil + 1}^{\infty} \Pr[L = \ell] c^{2\ell}.$$

La première est dans $O(M_0^2)$ et la deuxième est bornée supérieurement par

$$\sum_{\ell=\lceil \log_c M_0 \rceil + 1}^{\infty} (1 - P_{M_0})^{\ell - \lceil \log_c M_0 \rceil - 1} c^{2\ell} \leq \sum_{\ell=1}^{\infty} M_0^2 ((1 - P_{M_0}) c^2)^{\ell}.$$

Cette série converge car $(1 - P_M)c^2 < (1 - P_{M_0})c^2 = 1$ pour $M > M_0$. Elle est donc dans $O(M_0^2) = O(1/a)$. □

4.4 Autre travaux

Cette section offre des résultats préliminaires découlant de réflexions auxquels s'est adonné l'auteur et qui n'ont pas porté fruits, mais qui pourraient aider toute personne souhaitant reprendre ces travaux.

4.4.1 Une borne inférieure sur la variance

Nous avons borné supérieurement la variance de l'algorithme d'amplification de l'amplitude dans le cas où $c < \sqrt{2}$. Il serait intéressant d'avoir une borne inférieure sur la variance. Pour ce faire, il nous faut une borne inférieure sur la probabilité de s'arrêter à l'itération ℓ, $\Pr[L = \ell]$.

Dans les prochaines lignes, on tente de trouver une borne inférieure sur la probabilité de dépasser l'itération ℓ. Remarquons d'abord que

$$\begin{aligned} Pr[L > \ell] &= \prod_{i=1}^{\ell} \frac{1}{2}\left(1 + \frac{\sin(4c^i\theta)}{2c^i \sin(2\theta)}\right) \\ &\geq \prod_{i=1}^{\ell} \frac{1}{2}(1 + \cos(4c^i\theta)) \\ &\geq \prod_{i=1}^{\ell} (1 - 2c^i\theta) \end{aligned}$$

car $\sin(M\theta) \geq M\sin(\theta)\cos(M\theta)$ pour $M \geq 0$ et $0 \leq M\theta < \pi/2$. Cette dernière expression est un polynôme $p_\ell(\theta)$ de degré ℓ en θ. La propriété suivante de ce polynôme nous permettra de borner inférieurement la probabilité que la variable aléatoire L dépasse une certaine valeur.

Propriété 26. *$p_\ell(\theta) \geq 1 - a_\ell\theta$ où $a_\ell = \sum_{i=1}^{\ell} 2c^i$.*

Démonstration. Nous prouvons cette propriété par induction sur le degré ℓ de $p(\theta)$. Pour $\ell = 1$, nous avons évidemment $p(\theta) = 1 - 2c\theta = 1 - a_1\theta$. Supposons l'énoncé du lemme

vrai pour ℓ. Alors,

$$\begin{aligned} p_{\ell+1}(\theta) &= \prod_{i=1}^{\ell+1}(1-2c^i\theta) \\ &= p_\ell(\theta)(1-2c^{\ell+1}\theta) \\ &\geq (1-a_\ell\theta)(1-2c^{\ell+1}\theta) \\ &= 1-a_\ell\theta-2c^{\ell+1}\theta+2a_\ell c^{\ell+1}\theta^2 \\ &\geq 1-(a_\ell+2c^{\ell+1})\theta \\ &= 1-a_{\ell+1}\theta \end{aligned}$$ □

Puisque $a_\ell = 2c(c^\ell-1)/(c-1)$, on a

$$\Pr[L > \ell] \geq 1 - \frac{2c(c^\ell-1)}{c-1}\theta.$$

Cette borne n'est pas très serrée ; elle n'est utile que lorsque θ est très petit et c n'est pas trop près de 1. Si on veut, par exemple, que cette valeur soit plus grande que $1/2$, on devra avoir $\theta < \frac{c-1}{4c(c^\ell-2)}$. De plus, lorsque ℓ devient plus grand que $\log_c \theta + 1$, cette borne inférieure est négative, et donc complètement inutile.

4.4.2 Constante multiplicative

Comme nous l'avons vu dans la section 4.3, le choix du paramètre c a un impact significatif sur la variance du nombre de requêtes effectuées par l'algorithme d'amplification de l'amplitude. Le but de cette sous-section est de convaincre le lecteur que ce choix a aussi un impact sur l'espérance du nombre de requêtes.

Considérons le cas extrême où $c = 1$. Ce n'est pas parmi les valeurs permises pour c, mais le comportement de l'algorithme s'apparentera à celui-ci lorsque c est très près de 1. Dans ce cas, à chaque itération de l'algorithme, on applique $j = 0$ itération de Grover et on mesure. La probabilité de mesurer une solution est alors

$$\sin^2(\theta) = a.$$

La variable aléatoire du nombre d'itérations suit alors à peu près la loi géométrique de paramètre a. L'espérance du nombre de requêtes est dans l'ordre de $1/a$. On perd donc tout avantage sur le classique.

À l'autre extrême, lorsque c est très près de 2, on a une très grande probabilité de dépasser de beaucoup le nombre optimal d'itérations. Ce qui se produit est que lors d'une itération de l'algorithme, on applique en moyenne $M/2$ itérations de Grover. On échoue à mesurer une solution avec probabilité à peu près $1/2$ et à l'itération suivante on applique en moyenne M itérations de Grover, puis $2M$, etc. Ce qui fait en sorte qu'avec probabilité environ $1/2$, le nombre moyen de requêtes double.

Pour voir plus formellement ce qui se produit, considérons l'espérance du nombre de requêtes. En utilisant la technique de la preuve du théorème 24, elle peut être bornée inférieurement à une constante près, pour n'importe quel $s \in \mathbb{N}$, par

$$\sum_{\ell=1}^{\infty} \left(\frac{2 - 1/s}{4} \right)^{\ell} c^{\ell}$$

Ainsi, pour $c > 2$ l'espérance est infinie. On peut donc se douter que l'espérance, en fonction de c, croit rapidement vers l'infini lorsque $c \to 2$.

Il existe toutefois des valeurs de c entre les deux extrêmes pour lesquelles la constante cachée est raisonnable. Par exemple, dans [BBHT98], les auteurs utilisent la constante $c = 8/7$ et obtiennent une constante multiplicative d'environ 4 dans la notation grand-O. Ceci laisse présager qu'il existe une valeur optimale pour c dans l'intervalle $]1, 2[$. Une telle valeur permettrait d'atteindre la valeur minimale pour la constante multiplicative associée à la notation grand-O de l'espérance. Il serait intéressant de voir si cette valeur se trouve dans l'intervalle où la variance est finie.

Chapitre 5

Conclusion

Au chapitre 3, nous avons présenté des testeurs de propriété quantiques améliorant les meilleurs algorithmes existant qui testent la linéarité et la symétrie de fonctions booléennes. Ces nouveaux algorithmes ont été généralisés pour tester si une fonction est un homomorphisme entre eux groupes abéliens finis et tester la quasi-symétrie de fonctions booléennes. Chacun de ces nouveaux testeurs utilise l'algorithme d'amplification de l'amplitude afin d'augmenter l'amplitude d'une certaine composante d'un état associé à la fonction reçue en entrée. Ces testeurs s'appuient sur la promesse voulant que cette fonction soit a la propriété, soit est loin de l'avoir. Ceci permet de borner inférieurement l'amplitude initiale de cette composante, et ainsi, à borner supérieurement le nombre de requêtes faites par l'algorithme d'amplification de l'amplitude.

Le tableau suivant fait la synthèse des nouveaux et anciens algorithmes pour les problèmes considérés dans ce chapitre. Les cellules contiennent, pour chaque problème, la complexité en nombre de requêtes pour les meilleurs algorithmes classiques et quantiques connus précédemment, et pour les nouveaux algorithmes présentés dans ce mémoire. La complexité en requête est en fonction du paramètre de distance ϵ et de la probabilité d'erreur δ.

Tableau 5.I – Testeurs de propriété décrits dans ce mémoire

Propriété	Classique	Quantique	Contribution
Linéarité	$O(\frac{1}{\epsilon}\log\frac{1}{\delta})$ [BLR93]	$O((\frac{1}{\epsilon})^{\frac{2}{3}}\log\frac{1}{\delta})$ [HA11]	$O(\sqrt{\frac{1}{\epsilon}}\log\frac{1}{\delta})$
Homomorphisme	$O(\frac{1}{\epsilon}\log\frac{1}{\delta})$ [BLR93]	-	$O(\sqrt{\frac{1}{\epsilon}}\log\frac{1}{\delta})$
Symétrie	$O(\frac{1}{\epsilon}\log\frac{1}{\delta})$ [MP09]	$O((\frac{1}{\epsilon})^{\frac{2}{3}}\log\frac{1}{\delta})$ [HA11]	$O(\sqrt{\frac{1}{\epsilon}}\log\frac{1}{\delta})$
Quasi-symétrie	$O(\frac{n}{\epsilon}\log\frac{n}{\delta})$ [MP09]	-	$O(\frac{n}{\sqrt{\epsilon}}\log\frac{n}{\delta})$

Dans le chapitre 4, nous avons d'abord décrit la variable aléatoire correspondant au nombre de requêtes effectuées par l'algorithme d'amplification de l'amplitude pour une amplitude initiale inconnue. Nous avons ensuite reproduit la preuve que cette variable aléatoire a une valeur espérée dans $O(\sqrt{N})$ où N est la taille du domaine de la fonction f pour laquelle nous cherchons une solution. Dans la section suivante, nous avons montré que la variance de cette variable aléatoire est infinie pour $c > \sqrt{2}$ où c est le paramètre augmenté exponentiellement dans l'algorithme d'amplification de l'amplitude. Dans la même section, nous avons aussi montré que lorsque $c < \sqrt{2}$, alors la variance est finie et dans l'ordre de N. Finalement, une borne inférieure sur la probabilité de dépasser un nombre donné d'itérations a été présentée en fin de chapitre.

5.1 Travaux futurs

Les algorithmes présentés dans le chapitre 3 utilisent tous l'algorithme d'amplification de l'amplitude d'une manière semblable. La méthode utilisée est assez générale pour que l'on se demande quelles sont les autres propriétés pour lesquelles cette méthode donne un testeur efficace. Mieux encore, existe-t-il une famille de propriétés pour laquelle cette méthode peut être utilisée ?

Le chapitre 4 étudie la variable aléatoire du nombre de requêtes que fait l'algorithme d'amplification de l'amplitude. Seulement en restreignant le paramètre qui est augmenté exponentiellement peut-on borner la variance. Peut-on concevoir un algorithme accomplissant la même tâche avec une variance moindre ? Il serait aussi intéressant d'analyser comment l'espérance se comporte en fonction du paramètre c de l'algorithme et de tenter de trouver une valeur optimale pour ce paramètre.

Bibliographie

[BBC+95] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter, *Elementary gates for quantum computation*, Physical Review A **52** (1995), no. 5, 3457--3467.

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp, *Tight bounds on quantum searching*, Fortschritte der Physik **46** (1998), no. 4-5, 493--505.

[BHMT02] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp, *Quantum amplitude amplification and estimation*, Quantum Computation and Information (Samuel J. Lomonaco Jr. and Howard E. Brandt, eds.), Contemporary Mathematics, vol. 305, American Mathematical Society, 2002, pp. 53--74.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld, *Self-testing/correcting with applications to numerical problems*, Journal of Computer and System Sciences **47** (1993), no. 3, 549 -- 595.

[BV97] E. Bernstein and U. Vazirani, *Quantum complexity theory*, SIAM Journal on Computing **26** (1997), no. 5, 1411--1473.

[Gro97] Lov K. Grover, *Quantum mechanics helps in searching for a needle in a haystack*, Physical Review Letters **79** (1997), no. 2, 325--328.

[HA11] Mark Hillery and Erika Andersson, *Quantum tests for the linearity and permutation invariance of boolean functions*, Physical Review A **84** (2011), no. 6, 062329.

[Hir04] Mika Hirvensalo, *Quantum computing*, 2e ed., Natural Computing Series, Springer, 2004.

[Hø99] Peter Høyer, *Conjugated operators in quantum algorithms*, Physical Review A **59** (1999), no. 5, 3280--3289.

[MP09] Krzysztof Majewski and Nicholas Pippenger, *Attribute estimation and testing quasi-symmetry*, Information Processing Letters **109** (2009), no. 4, 233--237.

[NC10] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information : 10th anniversary edition*, Cambridge University Press, 2010.

[Sho97] Peter W. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing **26** (1997), no. 5, 1484--1509.

[WHH06] N.A. Weiss, P.T. Holmes, and M. Hardy, *A course in probability*, Addison Wesley, 2006.